DOCUMENT RESUME

ED 243 652 SE 044 383

AUTHOR Beam, John; And Others
TITLE Computer Mathematics 1B. Curriculum Bulletin Number 82CBM33.
INSTITUTION Houston Independent School District, Tex.
PUB DATE 82
NOTE 192p.
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052)

EDRS PRICE MF01/PC08 Plus Postage.
DESCRIPTORS *Computer Science Education; Computer Software; Curriculum Guides; Educational Objectives; Learning Activities; *Mathematics Curriculum; *Mathematics Instruction; *Programing; Resource Materials; Secondary Education; *Secondary School Mathematics; Teaching Methods
IDENTIFIERS *BASIC Programing Language; Texas (Houston)

ABSTRACT

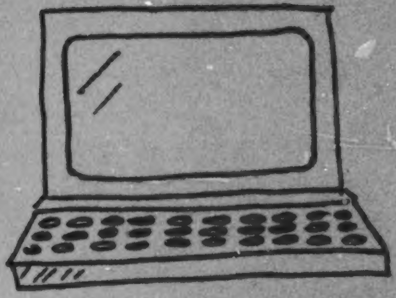
This guide for teachers in the Houston schools outlines the second semester of a two-semester course on computer mathematics. Course objectives are listed, and a scope-and-sequence chart correlates topics in the course with the textbook ("Computer Programming in the BASIC Language") and suggests vocabulary and time allotments. An objectives-by-topic cross-index is also included, and supplements are indexed to topics. The major portion of the guide presents plans for six objectives included in the textbook and a number of additional objectives not included in the textbook, to be taught if time allows. All provide activities aimed at teaching students how to program the computer for mathematics topics using the BASIC programing language. Appended are computer programs, plus a list of films and a bibliography. (MNS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

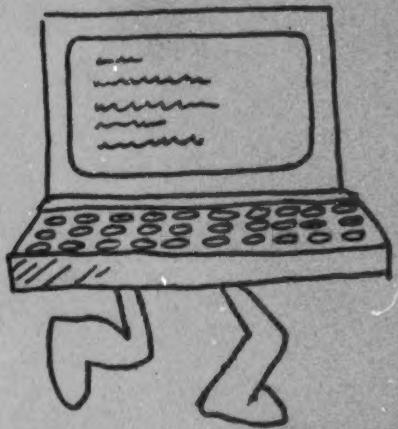
ED243652

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☑ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
● Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

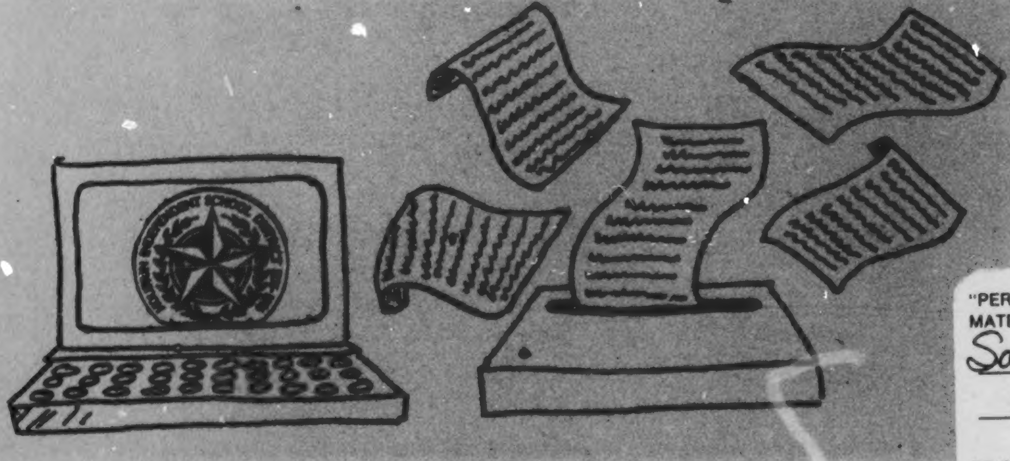
COMPUTER MATHEMATICS 18



see the computer



see the computer run



run, computer, run!

SE044383

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Sarah Abbott
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Houston Independent School District

Curriculum Bulletin Number 82CBM33

Item Number 33.0392

COMPUTER MATHEMATICS 1B

Curriculum Bulletin Number 82CBM33

Prepared by

John Beam, Bellaire High School
Carolyn Eubanks, Worthing High School
Elisabeth Jones, Sharpstown High School

Published by

Houston Independent School District
Houston, Texas
1982

Prepared Under the Direction of

Mrs. Carol Kuykendall, Assistant Superintendent, Curriculum
Ms. Madolyn Reed, Director, Mathematics Department
Mrs. Bettye Hall, Assistant Director, Secondary Mathematics
Mrs. Roberta Brisco, Assistant Director, Curriculum Publications

Board of Education

Mrs. Marquis L. (Blue) Alexander, Jr., President
Ms. Tina Reyes, Vice-President
Mr. Ray Morrison, Secretary
Mrs. Elizabeth Spates, Assistant Secretary
Mr. Tarrant Fendley
Mr. Wiley Henry
Dr. James C. Jones
Mrs. Cathy Mincberg
Mrs. Bobby Ann Peiffer

Administration

Dr. Billy R. Reagan, General Superintendent
Mrs. Patricia Shell, Superintendent for Instruction
Mr. Lawrence Marshall, Deputy Superintendent, General Instructional Services
and Alternative Education

PREFACE

This curriculum is designed for all computer teachers. Whether a first-year teacher or an "old pro," there are ideas to encourage and to challenge every teacher.

A few special notes to those teaching computer math for the first time:

1. Check the school library and audiovisual department at the beginning of the year for resource materials immediately available.
2. Class load should be dependent on equipment. Most schools do not have enough equipment to allow students to work independently. After the students have all had a chance to familiarize themselves with the equipment and classmates, allow them to sign up for programming groups. Stress that each student must contribute to the group in order to master the course objectives. Learning to work in groups is a beneficial experience for students because project teams are used in almost every phase of data processing. Groups should have a maximum of four members; two or three member groups would be preferable. Each group should have a specified amount of computer time (other groups will help monitor a group's time). As a general rule, each machine with printout and/or diskette capabilities should be able to handle three groups of three members each during a regular class period.
3. Students should meet minimum requirements as set up by H.I.S.D. The pre-requisite is one year of non-remedial mathematics.
4. An overhead projector will aid in the presentation of sample programs and instructions.
5. Supplementary material will be necessary to complete the course objectives in the guide.
6. Students should be encouraged to keep notebooks. These prove useful to them in the future in college courses and/or job situations.
7. Field trips and/or guest speakers are desirable. Students need to see large-scale processing. Oil companies, hospitals, and manufacturers of data processing equipment are usually willing to schedule tours, but call well in advance. Wait until students have become familiar with computer terminology before planning a trip.
8. At least 50% of a student's grade should be based on individual testing or projects. Group computer projects and daily exercises would comprise the other portion of a student's grade.

One final comment must be made concerning the programs in the appendix to the guide. In many cases an objective poses a problem which is answered by a reference to a BASIC program already in the text. In these cases the book's program is usually adequate to satisfy the objective. However, the appendix may also contain one or more programs that solve essentially the same problem, either by an alternate method or by one which is more efficient or elegant than that in the text. Such programs are included as reference for more advanced, eager student or for the teacher who may wish to examine alternatives and obtain ideas about programming efficiency and style. These programs should not be used as standards by which to judge the class. To do so might well prove discouraging both to the students and to the teacher. Some programs in the appendix, for example, matrix inversion for an arbitrary NxN matrix, are included for the possible later use by the teacher or student. It is not expected or necessarily desirable that the student or teacher trace through the programming logic in detail.

Table of Contents

Page

Preface iii
Scope and Sequence 1
Course Objectives 3
Vocabulary for Computer Mathematics B 5
Objectives-to-Topics Cross Index 6
List of Supplements 7
Suggested Activities 9
  Objective B4 11
    Topic 13 16
  Objective B4a* 26
    Topic 14 33
  Objective B6 34
  Objective B6a 52
    Topic 15 60
  Objective B7 61
  Objective B7a 64
  Additional Objectives 68
    Objective B2 69
    Objective B3 72
    Objective B8a 73
    Objective B9a 76
    Objective B10 88
    Objective B11a 94
    Objective B12a, B12b 98
    Objective B13a 101
    Objective B14 102
    Objective 14a 106
    Objective 15a, FORTRAN 116
    Objective B15a, COBOL 129
    Objective B16a 134
    Objective B17a 135
    Objective 17b, B17c 136
    Objective B17a(a) 138
    Objective B17a(b) 139
    Objective B17a(c) 140
    Objective B17a(d) 143
    Objective B17a(e) 144
    Objective B17a(f) 145
    Objective B17a(g) 146
    Objective B17a(i) 148
    Objective B17a(j) 149
Catalog of 16mm Computer Films 185
Bibliography 187

COMPUTER MATHEMATICS

PREREQUISITE: One non-remedial credit in Mathematics

CREDIT: 1/2, 1 credit

TEXTBOOK: Computer Programming in the BASIC Language
Jacobs, French, Moulds, and Schuchman
Allyn and Bacon, Inc.

AVERAGE COURSE SCOPE AND SEQUENCE*	TEXTBOOK CORRELATION	SUGGESTED TIME ALLOTMENT
COMPUTER MATHEMATICS A		
People and Computers	Topic 1	5 days
Elements of BASIC	Topic 2	5 days
Data In and Out	Topic 3	5 days
Flow Charts and Program Branching	Topic 4	10 days
For-Next Loops	Topic 5	10 days
Accumulating and Table Heading	Topic 6	5 days
BASIC Functions	Topic 7	10 days
Subscripting	Topic 8	10 days
Nested Loops and Sorting	Topic 9	12 days
Alphameric Output and Input	Topic 9	5 days
Term Project**	Obj. A16	8 days
Review and Final		2 days
		87 days

AVERAGE COURSE SCOPE AND SEQUENCE*	TEXTBOOK CORRELATION	SUGGESTED TIME ALLOTMENT
COMPUTER MATHEMATICS B		
Two-Dimensional Arrays	Topic 11	10 days
Functions and Subroutines	Topic 12	10 days
Interbase Conversions and Computer Codes	Topic 13	8 days
Matrix Algebra	Topic 14	10 days
Converging Processes	Topic 15	10 days
Optional Objectives	None	20 days
Term Project**	Obj. 12b	15 days
Review and Final		2 days
		85 days

*Computer Mathematics is best taught through flexible scheduling of class time. Topic I can be spread throughout the course. Some teachers may wish to begin flow charting immediately. The pace a teacher sets depends not only upon the preparation and aptitude of the students but also upon the number of students in the class and the number of computer terminals available. Some classes require two semesters to complete the 14 topics and others may finish in one semester, and can devote more time to the optional objectives.

**When assigning term projects a teacher must take into account the mathematical background and maturity of the individual student. A wide variety of topics should be presented from which the student may choose a single topic for a project.

COMPUTER MATHEMATICS B

COURSE OBJECTIVES

By the end of the second semester the student will:

① Demonstrate computer literacy beyond the level of Computer Mathematics A by being able to recognize and use in proper context the vocabulary designated in the curriculum guide.

② Demonstrate how integers can be stored in memory more compactly than real numbers.

③ Describe possible pitfalls when doing computer arithmetic and methods for avoiding them.

④ Convert numbers from each of the following bases to the other three: binary, octal, decimal, and hexadecimal.

4a. Use binary and hexadecimal notations to express characters in ASCII or EBCDIC.

⑤ Write BASIC programs which utilize user-defined function and subroutines.

⑥ Write BASIC programs which utilize MAT commands, where available, and which add, subtract, and multiply matrices without the use of the MAT command. (This requires the use of double subscripts.)

6a. Write a BASIC program to determine the value of a rank-N determinant.

6a*. Write a BASIC program to determine the inverse of a non-singular matrix.

⑦ Write BASIC programs which use a converging loop and appropriate iterative function to determine the square root of a positive real number and/or the cube root of a positive or negative real number.

7a. Write BASIC programs which use a converging loop and appropriate iterative functions to determine all real roots of a quadratic, a cubic, a quartic, or a quintic equation.

8a. Write BASIC programs using high-resolution graphics and any other special features available on the system.

9a. Write BASIC programs using data files, sequential files, and random files.

⑩ Solve a variety of mathematical, scientific and business application problems appropriate to the preparation and interests of the individual student.

11a. Write a simple assembly language program.

12a. Write programs involving at least three of the following: simulation, probability, decision trees, linear and/or binary searches, game theory, computer assisted instruction, Boolean Algebra and logical operators, cryptanalysis, three-dimensional representations and manipulations, pattern recognition.

12b. Expand one of the topics of 12a into a term project.

13a. Write BASIC programs using the PEEK function and the POKE command.

14 Write BASIC programs using the string functions LEFT$, RIGHT$, MID$, STR$, LEN, VAL, ASC, AND CHR$.

14a. Write a program which will teach and/or test a short, self-contained topic in any field.

15a. Write programs using other programming languages available on the system used, with emphasis on FORTRAN, COBOL, and/or PASCAL.

16a*. Expand data processing and elementary programming to a person not registered in Computer Mathematics, including administrators and teachers; write and demonstrate a program of interest to that person.

17 By the end of the second semester the student should have sufficient programming experience that he will be able to write BASIC programs which will

(a) Find the sum of two fractions, express the sum as a fraction, reduce the sum to its lowest terms, and print out the fractions and their sum.

(b) Solve, when coupled with a subroutine evaluating a rank-N determinant, a system of four linear equations .

17a. By the end of the second semester the more advanced student should also be able to write programs that will

(a) Find the sum or difference of two fractions input into the computer in normal algebraic form (for example, 2/3 - 3/7), express the sum or difference as a fraction reduced to lowest terms; and as output, print the fractions and their sum or difference in normal algebraic form (for example, 2/3 - 3/7 = 5/21).

(b) Find one real and two complex cube roots of a real number.

(c) Add, subtract, or multiply two real numbers of up to 30 digits.

(d) Print a calendar for any given year. (This program could be used as a project.)

(e) Compute n! and compare the exact value to Stirling's approximation.

(f) Find the zeroes of a polynomial by using Newton's method.

(g) Find the area under a curve by using Simpson's rule.

(h) Produce a payroll, an inventory, or solve some banking problem.

(i) Generate and list prime numbers by using the sieve method.

(j) Use a sorting procedure more efficient than the bubble sort to sort (alphabetize) numerical (alphabetical) data. (Two common methods are Shellsort and Quicksort.)

Vocabulary for Computer Mathematics B

complex number, pure imaginary number

convergent process, iteration

determinant, matrix, matrix inversion

field, file, sequential file, random file

key memory location, pointer

round-off error

subprogram, subroutine

Objectives-to-Topics Cross Index

Objective	Location Where Discussed
B1	Topics 12, 13, 14; Add'l. Obj. A17a(c), B
B2	Add'l. Obj. B2
B3	Add'l. Obj. B3, Prog. Append
B4	Topic 13, B14a; Prog. Append
B5	Topic 12
B6	Topics 11, 14; Prog Append
B6a*	Topic 14, Prog. Append
B6b	Topic 14, Prog. Append
B7	Topic 15
B7a	Topic 15
B8a	Add'l. Obj. B8a
B9a	Add'l. Obj. B9a, Prog. Append
B10	Add'l. Obj. B10
B11a	Add'l. Obj. B11a
B12a	Add'l. Obj. B12a
B12b	Add'l. Obj. B12a
B13a	Add'l. Obj. B13a, Prog. Append
B14	Add'l. Obj. B14, Prog. Append
B14a	Add'l. Obj. B14a
B15a	Add'l. Obj. B15a, Prog. Append B17a(g)
B16a*	Add'l. Obj. B16a*
B17(a)	Add'l. Obj. B17(a)
B17(b)	Add'l. Obj. B17(b)
B17(c)	Add'l. Obj. B17(c)
B17a(a)	Add'l. Obj. B17a(a), Prog. Append
B17a(b)	Add'l. Obj. B17a(b), Prog. Append
B17a(c)	Add'l. Obj. B17a(c), Prog. Append
B17a(d)	Add'l. Obj. B17a(d)
B17a(e)	Add'l. Obj. B17a(e), Prog. Append
B17a(f)	Add'l. Obj. B17a(f), Prog. Append
B17a(g)	Add'l. Obj. B17a(g), Prog. Append
B17a(h)	Add'l. Obj. B10
B17a(i)	Add'l. Obj. B17a(i), Prog. Append
B17a(j)	Add'l. Obj. B17a(j), Prog. Append

LIST OF SUPPLEMENTS, INDEXED TO TOPICS

Topic	Supplement	Title
13	Interbase Conversion and Computer Codes (Objectives B4 and B4a*)	
	B4.1	Computers and Binary Arithmetic
	B4.2	Worksheet: Binary and Decimal Numbers
	B4.3	Octal and Hexadecimal Number Systems
	B4.4	Sample Interbase Conversion Table
	B4.5	Worksheet: Converting to and from Decimal Numbers
	B4.6	Worksheet: Binary, Octal, and Hexadecimal Conversion
	B4.7	Codes: ASCII and EBCDIC
14	Matrix Algebra (Objectives B6, B6a, B6a*)	
	B6.1	Addition and Subtraction of Matrices
	B6.2	Multiplication of Matrices
	B6.3	Additional Matrix Problems
	B6a	Evaluation of Determinants
	B6a*	Matrix Inversion
15	Converging Process (Objectives B7, B7a)	
	None	
Additional Objectives - B [Objective B2, B3, B9a, B10, B11a, B12a, B12b, B13a, B14, B14a, B15a, B16a*, B17(a), B17(b), B17(c), B17a(a) through B17a(j)]		
	B2	Computer Representation of Integers and Real Numbers
	B8a	Graphics for the TRS-80 Microcomputer
	B9a.1	Introduction to Files
	B9a.2	Data File Assignment
	B9a.3	Quiz over BASIC Files
	B10	Sample Business Application Programs
	B11a	Cardiac Assembly Language

SUGGESTED ACTIVITIES

COMPUTER MATHEMATICS

Objective B4: Students will convert numbers from each of the following bases to the other three: decimal, binary, octal and hexadecimal.

Reference: Poirot, James L. and Groves, David N. *Computers and Mathematics*. pp. 269 - 306

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Go over the reasons for studying binary arithmetic.	Supplement B4.1
Methods of Converting from Decimal Numerals to Binary Numerals.	Supplement B4.2

$75_{10} = \underline{\hspace{3cm}}_2$

Division	Quotient	Remainder
$\frac{75}{2}$ =	37	1
$\frac{37}{2}$ =	18	1
$\frac{18}{2}$ =	9	0
$\frac{9}{2}$ =	4	1
$\frac{4}{2}$ =	2	0
$\frac{2}{2}$ =	1	0
$\frac{1}{2}$ =	0	1

Reverse the order of remainders and

$75_{10} = \underline{1001011}_2$

Objective B.4: (con't)

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Method of Converting from Binary Numerals to Decimal Numerals. $11011001_2 = \underline{\qquad\qquad}_{10}$ Use the powers of two to convert: $11011001_2 =$ $1(2^7) + 1(2^6) + 0(2^5) + 1(2^4) + 1(2^3) + 0(2^2) + 0(2^1) + 1(2^0) =$ $1(128) + 1(64) + 0(32) + 1(16) + 1(8) + 0(4) + 0(2) + 1(1) =$ $128 + 64 + 0 + 16 + 8 + 0 + 0 + 1 =$ 217_{10} $11011001_2 = \underline{217}_{10}$	Supplement B4.2
Discuss the purpose of octal and hexadecimal notation.	Supplement B4.3
Use the Table Comparing Bases for review of methods.	Supplement B4.4
Method of Converting from Decimal Numerals to Octal or Hexadecimal Numerals. Use the division remainder method as for binary. See the next page for examples.	

Objective B4: (con't)

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Decimal to Octal $2307_{10} = \underline{\qquad}_8$	Supplement B4.5

Division	Quotient	Remainder
$\frac{2307}{8} =$	288	3
$\frac{288}{8} =$	36	0
$\frac{36}{8} =$	4	4
$\frac{4}{8} =$	0	4

$2307_{10} = \underline{\ 4403\ }_8$

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Decimal to Hexadecimal $4323_{10} = \underline{\qquad}_{16}$	Supplement B4.5 Note: See Supplement B4.4 for hexadecimal notation.

Division	Quotient	Remainder
$\frac{4323}{16} =$	270	3
$\frac{270}{16} =$	16	14 or E
$\frac{16}{16} =$	1	0
$\frac{1}{16} =$	0	1

$4323_{10} = \underline{\ 10E3\ }_{16}$

Conversion from Octal or Hexadecimal to Decimal Numerals

Use powers and place value method as with binary numerals.

Objective B4: (con;t)

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Conversion from Binary to Octal or Hexadecimal. This conversion is based on the fact that both 8 and 16 are powers of 2. It is done by grouping the binary digits in groups (3 to convert to octal and 4 to convert to hexadecimal) beginning at the binary point. Examples: $110111_2 = \overline{110}\ \overline{111}_2 = 67_8$ $1010111_2 = \overline{001}\ \overline{010}\ \overline{111}_2 = 127_8$ $10110110_2 = \overline{1011}\ \overline{0110}_{16} = B6_{16}$ $10010111001_2 = \overline{0100}\ \overline{1011}\ \overline{1001}_2 = 4B9_{16}$	Supplement B4.6
Conversion from Octal and Hexadecimal to Binary. This conversion is done by writing the binary equivalents of each octal or hexadecimal digit. Zeros must be included to make the number of binary digits necessary (3 for each octal digit and 4 for each hexadecimal digit) Examples: $2341_8 = \overline{010}\ \overline{011}\ \overline{100}\ \overline{001}\ _2$ $74345_8 = \overline{111}\ \overline{011}\ \overline{100}\ \overline{101}\ \overline{000}\ _2$ $5C_{16} = \overline{0101}\ \overline{1100}\ _2$ $FAD_{16} = \overline{1111}\ \overline{1010}\ \overline{1101}\ _2$	Supplement B4.6

Objective B4: (con't)

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Conversion from Octal to Hexadecimal and from Hexadecimal to Octal. This is best done by converting to binary then converting to octal or hexadecimal. For a computer program which will convert between any two bases between 2 and 36, inclusive, see Program B4 in the Program Appendix. It should work on any computer which uses the ASC and CHR$ functions.	Supplement B4.6

TOPIC THIRTEEN: INTERBASE CONVERSION AND COMPUTER CODES

Performance Objectives:

The student will -

B4	Convert numbers from each of the following bases to the other three: decimal, binary, octal, and hexadecimal
B4a*	Use binary and hexadecimal notations to express characters in ASCII or EBCDIC
A1	Use appropriate computer vocabulary (ASCII or EBCDIC)

References:

Poirot, James L. and Groves, David N. Computers and Mathematics, pp. 269-306

Supplement B4.1

Computers and Binary Arithmetic

Our regularly used decimal system is based on ten digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). This system probably developed because of the fact that people have ten fingers (one digit per finger).

Computers, however, do not have fingers. Therefore, there is no logical reason for decimal arithmetic in computers. Computers are essentially electrical circuits controlled by switches. Instead of ten fingers, each switch has two positions - OFDF and ON represented by two digits (0, 1).

Any number can be represented with these two digits. Place value is used in both systems.

Decimal System

$$13069_{10} = 1\cdot(\text{ten}^4) + 3\cdot(\text{ten}^3) + 0\cdot(\text{ten}^2) + 6(\text{ten}^1) + 9(\text{ten}^0)$$

$$= 1\cdot(10_{10})^4 + 3\cdot(10_{10})^3 + 0\cdot(10_{10})^2 + 6(10_{10})^1 + 9(10_{10})^0$$

Binary System

$$1011011_{10} = 1\cdot(\text{two}^6) + 0\cdot(\text{two}^5) + 1\cdot(\text{two}^4) + 1\cdot(\text{two}^3) + 0\cdot(\text{two}^2) + 1(\text{two}^1) + 1(\text{two}^0)$$

$$= 1\cdot(10_2)^6 + 0\cdot(10_2)^5 + 1\cdot(10_2)^4 + 1\cdot(10_2)^3 + 0\cdot(10_2)^2 + 1\cdot(10_2)^1 + 1\cdot(10_2)^0$$

Key to Supplement B4.2

Binary and Decimal Numerals:

1. Decimal to Binary:

   a. $35_{10} = \underline{100011_2}$

   b. $28_{10} = \underline{11100_2}$

   c. $241_{10} = \underline{11110001_2}$

   d. $306_{10} = \underline{100110010_2}$

   e. $91_{10} = \underline{1011011_2}$

   f. $102_{10} = \underline{1100110_2}$

   g. $435_{10} = \underline{110110011_2}$

   h. $606_{10} = \underline{1001011110_2}$

   i. $1235_{10} = \underline{10011010011_2}$

   j. $5036_{10} = \underline{1001110101100_2}$

2. Binary to Decimal:

   a. $100_2 = \underline{\ \ 4}_{10}$

   b. $111_2 = \underline{\ \ 7}_{10}$

   c. $1011_2 = \underline{\ \ 11}_{10}$

   d. $11001_2 = \underline{\ \ 25}_{10}$

   e. $10001_2 = \underline{\ \ 17}_{10}$

   f. $101010_2 = \underline{\ \ 42}_{10}$

   g. $101111_2 = \underline{\ \ 47}_{10}$

   h. $1010011_2 = \underline{\ 83}_{10}$

   i. $11011111_2 = \underline{\ 223}_{10}$

   j. $111000110_2 = \underline{\ 454}_{10}$

Octal and Hexadecimal Number Systems

Binary numeration is used by computers. However, for purposes of visual represental binary notation can be bulky because of the length of numerals. One method used to get around this is the use of octal (base 8) and hexadecimal (base 16) notation.

Because 8 and 16 are powers of 2, converting between binary and either octal or hexadecimal notation is easily done.

Octal System

$$2073_8 = 2\cdot(8^3) + 0\cdot(8^2) + 7(8') + 3(8^0)$$

Hexadecimal System

Note: In "normal" decimal arithmetic, we need (and have) ten digits; however, for hexadecimal arithmetic we must have sixteen digits. We will simply include six extra digits and end up with 0, 1, 2, 3, 4, 5, 6, 7, 7, 8, 9, A, B, C, D, E, F.

$$\begin{aligned} 3F6B_{16} &= 3\cdot(16^3) + 15\cdot(16^2) + 6\cdot(16') + 11\cdot(16^0) \\ &= 3\cdot(10_{16})^3 + F\cdot(10_{16})^2 + 6\cdot(10_{16})^1 + B\cdot(10_{16})^0 \end{aligned}$$

Supplement B4.4

Sample Interbase Conversion Table

Base 10	Binary	Octal	Hexadecimal
0	000000	0	0
1	000001	1	1
2	000010	2	2
3	000011	3	3
4	000100	4	4
5	000101	5	5
6	000110	6	6
7	000111	7	7
8	001000	10	8
9	001001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

WORKSHEET: CONVERTING-TO-AND-FROM DECIMAL NUMERALS

1. Convert these decimal numerals to the indicated bases.

a. 34_{10} = __________

b. 29_{10} = __________

c. 307_{10} = __________

d. 489_{10} = __________

e. 367_{10} = __________

f. 509_{10} = __________

g. 986_{10} = __________

h. 2308_{10} = __________

i. 4936_{10} = __________

j. 23058_{10} = __________

2. Convert these numerals to base ten.

a. 43_8 = __________

b. 76_8 = __________

c. 203_8 = __________

d. 1470_8 = __________

e. 34051_8 = __________

f. 23_{16} = __________

g. $\mathrm{E\ F}_{16}$ = __________

h. $\mathrm{20B}_{16}$ = __________

i. 305_{16} = __________

j. IABC_{16} = __________

KEY TO
Supplement B4.5

WORKSHEET: CONVERTING-TO-AND-FROM DECIMAL NUMERALS

1. Convert these decimal numerals to the indicated bases.

a. $34_{10} = \underline{42}_{8}$

b. $29_{10} = \underline{1D}_{16}$

c. $307_{10} = \underline{463}_{8}$

d. $489_{10} = \underline{751}_{8}$

e. $367_{10} = \underline{16F}_{16}$

f. $509_{10} = \underline{111111101}_{2}$

g. $986_{10} = \underline{1732}_{8}$

h. $2308_{10} = \underline{904}_{16}$

i. $4936_{10} = \underline{11510}_{8}$

j. $23058_{10} = \underline{5A12}_{16}$

2. Convert these numerals to base ten.

a. $43_{8} = \underline{35}_{10}$

b. $76_{8} = \underline{62}_{10}$

c. $203_{8} = \underline{131}_{10}$

d. $1470_{8} = \underline{824}_{10}$

e. $34051_{8} = \underline{14377}_{10}$

f. $23_{16} = \underline{35}_{10}$

g. $E\ F_{16} = \underline{239}_{10}$

h. $20B_{16} = \underline{523}_{10}$

i. $305_{16} = \underline{773}_{10}$

j. $IABC_{16} = \underline{6844}_{10}$

WORKSHEET: BINARY, OCTAL, AND HEXADECIMAL CONVERSIONS

Convert to indicated form:

a. $110101_2 = \underline{\qquad\qquad}_8$

b. $110101_2 = \underline{\qquad\qquad}_{16}$

c. $1010001010_2 = \underline{\qquad\qquad}_8$

d. $1010001010_2 = \underline{\qquad\qquad}_{16}$

e. $111111_2 = \underline{\qquad\qquad}_8$

f. $111111_2 = \underline{\qquad\qquad}_{16}$

g. $10010000_2 = \underline{\qquad\qquad}_8$

h. $10010000_2 = \underline{\qquad\qquad}_{16}$

i. $101010101010111_2 = \underline{\qquad\qquad}_8$

j. $101010101010111_2 = \underline{\qquad\qquad}_{16}$

k. $43_8 = \underline{\qquad\qquad}_2$

l. $45_8 = \underline{\qquad\qquad}_2$

m. $23_{16} = \underline{\qquad\qquad}_2$

n. $3E_{16} = \underline{\qquad\qquad}_2$

o. $F0_{16} = \underline{\qquad\qquad}_2$

p. $2035_8 = \underline{\qquad\qquad}_2$

q. $4203_8 = \underline{\qquad\qquad}_2$

r. $20007_8 = \underline{\qquad\qquad}_2$

s. $3EF_{16} = \underline{\qquad\qquad}_2$

t. $2D03C_{16} = \underline{\qquad\qquad}_2$

u. $43_8 = \underline{\qquad\qquad}_{16}$

v. $B0_{16} = \underline{\qquad\qquad}_8$

w. $321_8 = \underline{\qquad\qquad}_{16}$

x. $431_{16} = \underline{\qquad\qquad}_8$

y. $2012_8 = \underline{\qquad\qquad}_{16}$

KEY TO
SUPPLEMENT B4.6

WORKSHEET: BINARY, OCTAL, AND HEXADECIMAL CONVERSIONS

Convert to indicated form:

a. $\overline{110}\;\overline{101}_2 = \underline{\quad 65 \quad}_8$

b. $\overline{110}\;\overline{101}_2 = \underline{\quad 35 \quad}_{16}$

c. $\overline{1}\;\overline{010}\;\overline{001}\;\overline{010}_2 = \underline{\quad 1212 \quad}_8$

d. $\overline{10}\;\overline{1000}\;\overline{1010}_2 = \underline{\quad 28A \quad}_{16}$

e. $\overline{111}\;\overline{111}_2 = \underline{\quad 77 \quad}_8$

f. $11\;\overline{1111}_2 = \underline{\quad 3F \quad}_{16}$

g. $\overline{10}\;\overline{010}\;\overline{000}_2 = \underline{\quad 220 \quad}_8$

h. $\overline{1001}\;\overline{0000}_2 = \underline{\quad 90 \quad}_{16}$

i. $\overline{101}\;\overline{010}\;\overline{101}\;\overline{010}\;\overline{111}_2 = \underline{\quad 52527 \quad}_8$

j. $\overline{101}\;\overline{0101}\;\overline{0101}\;\overline{0111}_2 = \underline{\quad 557 \quad}_{16}$

k. $43_8 = \underline{\quad \overline{100}\;\overline{011} \quad}_2$

l. $45_8 = \underline{\quad 100\;101 \quad}_2$

m. $23_{16} = \underline{10}_2$

n. $3E_{16} = \underline{\quad 11\;1110 \quad}_2$

o. $F0_{16} = \underline{\quad 1111\;0000 \quad}_2$

p. $2035_8 = \underline{10\;000\;011\;101 \quad}_2$

q. $4203_8 = \underline{100\;010\;000\;011 \quad}_2$

r. $20007_8 = \underline{\quad 10\;000\;000\;000\;111 \quad}_2$

s. $3Ef_{16} = \underline{\quad 11\;1110\;1111 \quad}_2$

t. $2D03C_{16} = \underline{10\;1101\;0000\;0011\;1100}_2$

u. $43_8 = \underline{100\;011}_2 = \underline{23}_{16}$

v. $B0_{16} = \underline{1011000}_2 = \underline{260}_8$

w. $321_8 = \underline{11010001}_2 = \underline{D1}_{16}$

x. $431_{16} = \underline{10000110001}_2 = \underline{2061}_8$

y. $2012_8 = \underline{10000001010}_2 = \underline{40A}_{16}$

COMPUTER MATHEMATICS

Objective: B4a* - Use binary and hexadecimal notations to express characters in ASCII and EBCDIC code.
A1 - Computer Vocabulary: ASCII, EBCDIC

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Discuss the need for codes. (Supplement B4.7)	Assignment: Supplement B4.7
Use either ASCII or EBCDIC as an example of binary codes. Go over examples of conversions. (Supplement B4.7) (Table Supplement B4.7)	
Review need and examples of hex. (Supplement B4.7)	
Explain that this is a very simplified version of EBCDIC. When actually used, EBCDIC includes other things such as parity or check bits. However, if the simplified version is understood now, the other aspects can easily be added when needed.	

Supplement B4.7

Codes - ASCII and EBCDIC

When programs are run and data are stored by computer, numbers are only one type of information included. There are also letters, operations, punctuation, and other symbols. The term alphanumeric refers to all of these. The computer, however, "recognizes" only the binary digits 0 and 1, for OFF and ON. Therefore, all information must be converted to binary form to be used by the computer.

Alphanumeric data are represented in different computer systems by different binary codes. Two of the most commonly used codes are EBCDIC (Extended Binary Coded Decimal Interchange Code) and ASCII (American Standard Code for Information Interchange).

With an understanding of the basic method of binary codes and the use of tables, it is possible to express alphanumeric data in any binary code. EBCDIC will be used as an example of these codes. (Table Supplement B 4.7)

EBCDIC: Extended Binary Coded Decimal Interchange Code

The Extended Binary Coded Decimal Interchange Code was developed by IBM for use by its computer systems. It is now used world-wide. It is based on a table in which each decimal digit, letter, punctuation, or other special character is represented by an eight-bit unit. Each bit is a single binary digit, either 0 or 1. Each eight-bit unit or character representation is called a byte.

To change a word or number into EBCDIC, take each letter of digit, find its EBCDIC representation in the table, then put them together.

Examples:

Convert to EBCDIC

a) 7 0 3

$$\overbrace{1111\ 0111}^{7}\quad\overbrace{1111\ 0000}^{0}\quad\overbrace{1111\ 0011}^{3}$$

b) C O D E

$$\overbrace{1100\ 0011}^{C}\quad\overbrace{1101\ 0110}^{O}\quad\overbrace{1100\ 0100}^{D}\quad\overbrace{1100\ 0101}^{E}$$

c) If A < 5 then 25

$$\overbrace{1100\ \ 0001}^{I}\quad\overbrace{1100\ \ 0110}^{F}\quad\overbrace{1100\ \ 0001}^{A}\qquad\overbrace{0100\ \ 1100}^{<}\qquad\overbrace{1111\ \ 0101}^{5}$$

$$\overbrace{1110\ \ 0011}^{T}\quad\overbrace{1100\ \ 1001}^{H}\quad\overbrace{1100\ \ 0101}^{E}\quad\overbrace{1101\ \ 0101}^{N}\quad\overbrace{1111\ \ 0010}^{2}\quad\overbrace{1111\ \ 0101}^{5}$$

To change from EBCDIC to normal characters, divide the binary representation into eight-bit bytes and then convert each byte to the appropriate character.

Examples:

Convert each set of EBCDIC code characters to words or numbers or statements.

a) $\underbrace{1111\ 0001}_{1}\ \underbrace{1111\ 0010}_{2}\ \underbrace{1111\ 1001}_{9}$ b) $\underbrace{1100\ 1000}_{H}\ \underbrace{1100\ 0101}_{E}\ \underbrace{1101\ 0011}_{L}\ \underbrace{1101\ 0111}_{P}$

c) $\underbrace{1100\ 0111}_{G}\ \underbrace{1101\ 0110}_{O}\ \underbrace{1110\ 0011}_{T}\ \underbrace{1101\ 0110}_{O}\ \underbrace{1111\ 1000}_{8}\ \underbrace{1111\ 0110}_{6}\ \underbrace{1111\ 0100}_{4}$

HEX: SHORTHAND NOTATION FOR EBCDIC

Since EBCDIC notation with its 8 bits per character can be clumsy and difficult to read, a "shorthand" has been applied to it. This shorthand system is based on the hexadecimal (base 16) numeration. Each eight-bit binary byte can be expressed as two hexadecimal characters. This greatly simplifies reading of EBCDIC notation. The EBCDIC table (Supplement B4) includes the represented character, EBCDIC binary notation, and Hex notation.

Example:

Characters: COMPUTER

EBCDIC Binary : $\overbrace{1100\ 0011}^{C}\ \overbrace{1100\ 0110}^{O}\ \overbrace{1100\ 0100}^{M}\ \overbrace{1100\ 0111}^{P}$

$\overbrace{1110\ 0100}^{U}\ \overbrace{1110\ 0011}^{T}\ \overbrace{1100\ 0101}^{E}\ \overbrace{1101\ 1001}^{R}$

Hex: $\overbrace{C3}^{C}\ \overbrace{D6}^{O}\ \overbrace{D4}^{M}\ \overbrace{D7}^{P}\ \overbrace{E4}^{U}\ \overbrace{E3}^{T}\ \overbrace{C5}^{E}\ \overbrace{D9}^{R}$

WORKSHEET: EBCDIC

1) Convert to EBCDIC binary code and hex notation

a) PRINT b) MATHEMATICS

c) BINARY CODE

2) Convert from EBCDIC binary to hex notation and to letters or numbers.

a) 1100 0101 1100 0010 1100 0011 1100 0100 1100 1001 1100 0011

b) 1100 0011 1101 0110 1101 0101 1110 0101 1100 0101 1101 1001
1110 0010 1100 1001 1101 0110 1101 0101

c) 1110 0011 1101 0110 1100 1000 1100 0101 1110 0111 1100 0001
1100 0100 1100 0101 1100 0011 1100 1001 1101 0100 1100 0001 1101 0011

3) Convert from hex rotation to EBCDIC binary code and to letters or numbers.

a) F4 F3 F5 F6 F8

b) C9 E2 D4 E4 D3 E3 C9 D7 D3 C5

c) D6 C6 F8

KEY
SUPPLEMENT B4.7

WORKSHEET: EBCDIC

1. a) 1101 0111 1101 1001 1100 1001 1101 0101 1110 0011

D7 D9 C9 D5 E3

b) 1101 0100 1100 0001 1110 0011 1100 1000 1100 0101 1101 0100
1100 0001 1110 0011 1100 1001 1100 0011 1110 0010

D4 C1 E3 C8 C5 D4 C1 E3 C9 C3 F2

c) 1100 0010 1100 1001 1101 0101 1100 0001 1101 1001 1110 1000
1100 0011 1101 0110 1100 0100 1100 0101

C2 C9 D5 C1 D9 E8 C3 D6 C4 C5

2. a) C5 C2 C3 C4 C9 C3
E B C D I C

b) C3 D6 D5 E5 C5 D9 E2 C9 D6 D5
C O N V E R S I O N

c) E3 D6 C8 C5 E7 C1 C4 C5 C3 C9 D4 C1 D3
T O H E X A D E C I M A L

3. a) 1111 0100 1111 0011 1111 0101 1111 0110 1111 1000
4 3 5 6 8

b) 1100 1001 1110 0010 1101 0100 1110 0100 1101 0011
1110 0011 1100 1001 1101 0111 1101 0011 1100 0110

IS MULTIPLE

c) 1101 0110 1100 0110 1111 1000
0 F 8

PARTIAL TABLE OF ASCII CHARACTER ASSIGNMENTS

Character	ASCII base 10	ASCII binary	Character	ASCII base 10	ASCII binary
SPACE	32	010 0000	@	64	100 0000
!	33	010 0001	A	65	100 0001
"	34	010 0010	B	66	100 0010
#	35	010 0011	C	67	100 0011
$	36	010 0100	D	68	100 0100
%	37	010 0101	E	69	100 0100
&	38	010 0110	F	70	100 0110
'	39	010 0111	G	71	100 0111
(	40	010 1000	H	72	100 1000
)	41	010 1001	I	73	100 1001
*	42	010 1010	J	74	100 1010
+	43	010 1011	K	75	100 1011
,	44	010 1100	L	76	100 1100
—	45	010 1101	M	77	100 1101
.	46	010 1110	N	78	100 1110
/	47	010 1111	O	79	100 1111
0	48	011 0000	P	80	101 0000
1	49	011 0001	Q	81	101 0001
2	50	011 0010	R	82	101 0010
3	51	011 0011	S	83	101 0011
4	52	011 0100	T	84	101 0100
5	53	011 0101	U	85	101 0101
6	54	011 0110	V	86	101 0110
7	55	011 0111	W	87	101 0111
8	56	011 1000	X	88	101 1000
9	57	011 1001	Y	89	101 1001
:	58	011 1010	Z	90	101 1010
;	59	011 1011	[	91	101 1011
<	60	011 1100	\	92	101 1100
=	61	011 1101	]	93	101 1101
>	62	011 1110	↑	94	101 1110
?	63	011 1111	-	95	101 1111

Lower-case letter symbols are coded by adding 32 to the upper-case letters. Thus

Character	ASCII
a	97 or 110 0000
b	98 or 110 0010
y	115 or 111 100
z	116 or 111 1000

PARTIAL TABLE OF EBCDIC CHARACTER ASSIGNMENTS

Character	EBCDIC Binary	Hex Shorthand	Character	EBCDIC Binary	Hex Shorthand	Character	EBCDIC Binary	Hex Shorthand
A	1100 0001	C1	a	1000 0001	81	0	1111 0000	F0
B	1100 0010	C2	b	1000 0010	82	1	1111 0001	F1
C	1100 0011	C3	c	1000 0011	83	2	1111 0010	F2
D	1100 0100	C4	d	1000 0100	84	3	1111 0011	F3
E	1100 0101	C5	e	1000 0101	85	4	1111 0100	F4
F	1100 0110	C6	f	1000 0110	86	5	1111 0101	F5
G	1100 0111	C7	g	1000 0111	87	6	1111 0110	F6
H	1100 1000	C8	h	1000 1000	88	7	1111 0111	F7
I	1100 1001	C9	i	1000 1001	89	8	1111 1000	F8
J	1101 0001	D1	j	1001 0001	91	9	1111 1001	F9
K	1101 0010	D2	k	1001 0010	92	Bell	0010 1111	2F
L	1101 0011	D3	l	1001 0011	93	Back Space	0001 0110	16
M	1101 0100	D4	m	1001 0100	94	Carriage Return	0000 0111	0D
N	1101 0101	D5	n	1001 0101	95			
O	1101 0110	D6	o	1001 0110	96	Delete	0000 0111	07
P	1101 0111	D7	p	1001 0111	97	End of Text	0011 0111	37
Q	1101 1000	D8	q	1001 1000	98	Escape	0010 0111	28
R	1101 1001	D9	r	1001 1001	99	_	0110 0000	60
S	1110 0010	E2	s	1001 0010	A2	/	0110 0001	61
T	1110 0011	E3	t	1010 0011	A3	$	0101 1011	5B
U	1110 0100	E4	u	1010 0100	A4	<	0100 1100	4C
V	1110 0101	E5	v	1010 0101	A5	*	0101 1100	5C
W	1110 0110	E6	w	1010 0110	A6	(	0100 1101	4D
X	1110 0111	E7	x	1010 0111	A7	)	0101 1101	5D
Y	1110 1000	E8	y	1010 1000	A8	+	0100 1110	4E
Z	1110 1001	E9	z	1010 1001	A9	>	0110 1110	6E
						=	0111 1110	7E

TOPIC FOURTEEN: MATRIX ALGEBRA

Performance Objectives:

The student will -

B6 Write BASIC programs which utilize MAT commands, where available; and which add, subtract, and multiply matrices without the use of the MAT commands.

B6a Write a BASIC program to determine the value of a rank-N determinant.

B6a* Write a BASIC program to determine the inverse of a non-singular matrix.

Reference:

Textbook: pages 389-429

COMPUTER MATHEMATICS

Objective: B6 - The student will write BASIC programs which utilize MAT commands, where available; and which add, subtract, and multiply matrices without the use of MAT commands.

Reference: Pages 389 - 397

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
First review matrices.	Textbook
(If Topic Eleven was not covered first semester, go back and pick it up before continuing with TOPIC THIRTEEN.)	Oral Exercise One, p. 397, #1-4
Be sure students can add and subtract matrices and that they understand what they are doing before doing it by computer.	Supplement B6.1
REMEMBER: Only matrices with identical dimensions can be added or subtracted.	
Multiplication of matrices may need to be introduced and taught. Careful review may be sufficient.	Textbook
Follow the steps in the textbook for vector and matrix multiplication.	Oral Exercise Two, pp. 404-405, #1-5 Objective Activity, pp. 415-417, #1-5
Go over matrix multiplication with and without MAT statements.	Written Problem Set Thirteen, pp. 423-426, #1-3
REMEMBER: Matrix multiplication may be performed on two matrices, only if the number of columns of the first equals the number of rows of the second.	Programs Problem Set Thirteen, pp. 426-427, #4-6 (The answers to #4 are not given in the answer key. They are: 4a - 5 4b 573 4c 12 0 - 7 -1 -2 8 15 4
The same product has the same number of rows as the first and the same number of columns as the second.	Supplement B6.2

TEACHING SUGGESTIONS

This last section is made up of examples of the use of matrices.

Go over them carefully.

Do traces of all programs.

Before using Supplement B6.3, discuss the transpose of a matrix.

Program B6 of the Program Appendix lists a program which multiplies any two matrices input by the user and prints the product matrix. An error message is given if matrix multiplication is not possible.

STUDENT ACTIVITIES

Textbook

Oral
Exercise Three, page 413, #-14

Objective Activity, pp. 418-419, #6-10

Programs
Problem Set Thirteen, p. 427, #7-8

Additional Problems: Supplement B6.3

ADDITION AND SUBTRACTION OF MATRICES

GIVEN:

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 6 & -1 & 8 \end{bmatrix} \qquad B = \begin{bmatrix} 4 & 3 & -6 \\ 1 & 2 & -8 \end{bmatrix}$$

$$S = A + B \quad \text{and} \quad D = A - B$$

1. Without using the computer, find S and D
2. Write and process a program using FOR-NEXT, double-subsrcipting. and any other necessary tools which will find S and D without using the MAT statements.
3. If MAT statements are available on the system, write and process a program using MAT statements to find S and D

ADDITION AND SUBTRACTION OF MATRICES

1. $S = \begin{bmatrix} 6 & -6 & -2 \\ 7 & 1 & 0 \end{bmatrix}$ $D = \begin{bmatrix} -2 & 0 & 10 \\ 5 & -5 & 16 \end{bmatrix}$

2.
```
10   REM MATRIX ADDITION AND SUBTRACTION WITHOUT MAT STATEMENTS
20   DIM A(2,3), B(2,3), S(2,3), D(2,3)
30   PRINT "FIRST MATRIX"
40   FOR I = 1 TO 2
50   FOR J = 1 TO 3
60   READ A(I,J)
70   PRINT A(I,J),
80   NEXT J
90   PRINT
100  NEXT I
110  PRINT
120  PRINT "SECOND MATRIX"
130  FOR I = 1 TO 2
140  FOR J = 1 TO 3
150  READ B(I,J)
160  PRINT B(I,J),
170  NEXT J
180  PRINT
190  NEXT I
200  PRINT
210  PRINT "MATRIX SUM"
220  FOR I = 1 TO 2
230  FOR J = 1 TO 3
240  LET S(I,J) = A(I,J) + B(I,J)
250  PRINT S(I,J),
260  NEXT J
270  PRINT
280  NEXT I
290  PRINT "MATRIX DIFFERENCE"
300  FOR I = 1 TO 2
310  FOR J = 1 TO 3
320  LET D(I,J) = A(I,J) - B(I,J)
```

```
2.(con't)
     330   PRINT D(I,J),
     340   NEXT J
     350   PRINT
     360   NEXT I
     370   DATA 2, 3, 4, 6, -1, 8
     380   DATA 4, 3, 6, 1, 2, -8
     390   END

3.    10   REM  MATRIX ADDITION AND SUBTRACTION WITH MAT STSTEMENTS
      20   DIM  A(2,3), B(2,3), S(2,3), D(2,3)
      30   PRINT "FIRST MATRIX"
      40   MAT READ A
      50   MAT PRINT A
      60   PRINT
      70   PRINT "SECOND MATRIX"
      80   MAT READ B
      90   MAT PRINT B
     100   PRINT
     110   PRINT "MATRIX SUM"
     120   MAT S = A + B
     130   MAT PRINT S
     140   PRINT
     150   PRINT "MATRIX DIFFERENCE"
     160   MAT D = A - B
     170   MAT PRINT D
     180   DATA 2, 3, 4, 6, -1, 8
     190   DATA 4, 3, 6, 1, 2, -8
     200   END
```

<u>OUTPUT</u> for #2 and #3 will be the same:

```
FIRST MATRIX

2    3    4

6   -1    8

SECOND MATRIX

4    3   -6

1    2   -8

MATRIX SUM

6    6   -2

7    1    0

MATRIX DIFFERENCE

-2    0    10

 5   -3    16
```

MULTIPLICATION OF MATRICES

GIVEN:

$$A = \begin{bmatrix} 2 & 3 & 5 \\ 6 & 1 & 0 \end{bmatrix} \quad \text{and } B = \begin{bmatrix} 0 & 3 & 2 & 1 \\ 1 & 1 & 4 & 0 \\ 2 & -2 & 3 & -1 \end{bmatrix}$$

P = A * B

1. Without using the computer, find P.

2. Write and process a program which will find P without using MAT statements.

3. Write and process a program using MAT statement to find P.

KEY to
Supplement B6.2

MULTIPLICATION OF MATRICES

1. Product

Matrix P = Matrix A * Matrix B

```
13   -1   31   -3
 1   19   16    6
```

2.

```
 10 REM MATRIX MULTIPLICATION WITHOUT MAT STATEMENTS
 20 DIM A(2,3),  B(3,4), P(2,4)
 30 PRINT "FIRST MATRIX"
 40 FOR I = 1 TO 2
 50 FOR J = 1 TO 3
 60 READ A (I,J)
 70 PRINT A(I.J)
 80 NEXT J
 90 PRINT
100 NEXT I
110 PRINT
120 PRINT "SECOND MATRIX"
130 FOR I = 1 TO 2
140 FOR J = 1 TO 3
150 READ B(I,J)
160 PRINT B(I,J)
170 NEXT J
180 PRINT
190 NEXT I
200 PRINT
210 PRINT "PRODUCT"
220 FOR I = 1 TO 2
230 FOR J = 1 TO 4
240 LET P(I,J) = 0
250 FOR M = 1 TO 3
260 LET P(I,J) = P(I,J) + A(I,M)*B(M,J)
270 NEXT M
280 PRINT P(I,J),
290 NEXT J
300 PRINT
310 NEXT I
320 DATA 2,3,5,6,1,0
330 DATA 0,3,2,1,1,1,4,0,2,-2,3,-1
340 END
```

3.
```
 10   REM MATRIX MULTIPLICATION WITH MAT STATEMENTS
 20   DIM A(2,3), B(3,4), P(2,4)
 30   PRINT "FIRST MATRIX"
 40   MAT READ A
 50   MAT PRINT A
 60   PRINT
 70   PRINT "SECOND MATRIX"
 80   MAT READ B
 90   MAT PRINT B
100   PRINT
110   PRINT "PRODUCT"
120   MAT P = A * B
130   MAT PRINT P
140   DATA 2, 3, 5, 6, 1, 0
150   DATA 0, 3, 2, 1, 1, 1, 4, 0, 2, -2, 3, -1
160   END
```

Output for #2 and #3 will be the same:

```
FIRST MATRIX

   2       3       5

   6       1       0

SECOND MATRIX

   0       3       2       1

   1       1       4       0

   2      -2       3      -1

PRODUCT

  13      -1      31      -3

  1       19      16       6
```

Supplement B6.3

ADDITIONAL MATRIX PROBLEMS

Below are printed two 5 x 5 matrices A and B. Write and process programs that will do the following:

1) a. Print A
   b. Print B
   c. Print A and B side by side

2) Print the matrix C which will be the sum of A and B

3) Print the matrix D which will be the difference of A and B (A-B)

4) Print the matrix E which will be the product of A and B

5) Print the transpose of A

6) Print the largest and smallest entry of E in 4.

$$A = \begin{matrix} 24 & 7 & 8 & 13 & 1 \\ 19 & 15 & 21 & 2 & 12 \\ 16 & 14 & 6 & 10 & 20 \\ 4 & 9 & 5 & 3 & 22 \\ 25 & 11 & 18 & 23 & 17 \end{matrix}$$

$$B = \begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 8 & 9 & 10 & 11 & 12 \\ 6 & 7 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 25 \\ 20 & 21 & 22 & 23 & 24 \end{matrix}$$

Partial programs for Supplement B6.3

1) a. To print A:

```
 10 FOR I = 1 TO 5
 20 FOR J = 1 TO 5
 30 READ A(I,J)
 40 PRINT A(I,J)
 50 NEXT J
 60 PRINT
 70 NEXT I
 80 DATA 24,7,8,13,1,19,15,21,2,12
 90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
100 END
```

b. To print B: Change these lines.

```
30 READ B(I,J)
40 PRINT B(I,J)
80 DATA 1,2,3,4,5,8,9,10,11,12
90 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
```

b. To print A and B side by side:

```
 10 FOR I = 1 TO 5
 20 FOR J = 1 TO 5
 30 READ A(I,J)
 40 NEXT J
 50 NEXT I
 60 FOR I = 1 TO 5
 70 FOR J = 1 TO 5
 75 READ B(I,J)
 76 NEXT J
 77 NEXT I
 80 DATA 24,7,8,13,1,19,15,21,2,12
 85 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
 90 DATA 1,2,3,4,5,8,9,10,11,12
 95 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
100 FOR I = 1 TO 5
110 FOR J = 1 TO 5
*120 IF A(I,J)<10 PRINT " ": A(I,J);: GO TO 140
130 PRINT A(I,J);
140 NEXT J
145 PRINT TAB(32);
150 FOR K = 1 TO 5
*151 IF B(I,K)<10  PRINT " ": B(I,K);: GO TO 170
160 PRINT B(I,K);
170 NEXTK
180 PRINT
190 NEXT I
200 END
```

*These statements line the numbers up evenly.

2) (See 1b program for steps 10-95)
The remainder of the program should be:

```
 100 FOR I = 1 TO 5
 110 FOR J = 1 TO 5
 120 C(I,J) = A(I,J) + B(I,J)
 130 IF C(I,J) <10 PRINT " ";: GO TO 140
 135 PRINT C(I,J)
 140 NEXT J
 150 PRINT
 160 NEXT I
 170 END
```

resulting matrix C: $\begin{bmatrix} 25 & 9 & 11 & 17 & 6 \\ 27 & 24 & 31 & 13 & 24 \\ 22 & 21 & 19 & 24 & 35 \\ 20 & 26 & 23 & 22 & 47 \\ 45 & 32 & 40 & 46 & 41 \end{bmatrix}$

3) (See 1b program for steps 10-95)
The remainder of the program should be:

```
 100 FOR I = 1 TO 5
*105 L = 0
 110 FOR J = 1 TO 5
 120 D(I,J) = A(I,J) - B(I,J)
*130 PRINT TAB(L); C(I,J);
*135 L = L + 5
 140 NEXT J
 150 PRINT
 160 NEXT I
 170 END
```

resulting matrix D: $\begin{bmatrix} 23 & 5 & 5 & 9 & -4 \\ 11 & 6 & 11 & -9 & 0 \\ 10 & 7 & -7 & -4 & 5 \\ -12 & -8 & -13 & -16 & -3 \\ 5 & -10 & -4 & 0 & -7 \end{bmatrix}$

*This provides another method of lining members up evenly.

4) (See 1b program for steps 10-95)
The remainder of the program should be:

```
100 FOR I = 1 TO 5
105 L = 0
110 FOR J = 1 TO 5
115 FOR K = 1 TO 5
120 E(I,J) = E(I,J) + A(I,K)*B(K,J)
130 NEXT K
135 PRINT TAB(L); E(I,J);
136 L = L + 7
140 NEXT J
150 PRINT
160 NEXT I
170 END
```

resulting matrix E:

$$\begin{bmatrix} 356 & 409 & 502 & 555 & 673 \\ 537 & 606 & 780 & 849 & 928 \\ 724 & 790 & 886 & 952 & 1068 \\ 594 & 637 & 705 & 748 & 806 \\ 929 & 1023 & 1207 & 1301 & 1510 \end{bmatrix}$$

5)

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR J = 1 TO 5
70 FOR I = 1 TO 5
80 PRINT A(I,J);
90 NEXT I
100 PRINT
110 NEXT J
120 DATA 24,7,8,13,1,19,15,21,2,12
125 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
130 END
```

$$\begin{bmatrix} 24 & 19 & 16 & 4 & 25 \\ 7 & 15 & 14 & 9 & 11 \\ 8 & 21 & 6 & 5 & 18 \\ 13 & 2 & 10 & 3 & 23 \\ 1 & 12 & 20 & 22 & 17 \end{bmatrix}$$

6) (See programs 1b and 4)
The remainder of the program should be:

```
170 X = E(1,1)
180 FOR I = 1 TO 5
190 FOR J = 1 TO 5
200 IF X >=E(I,J) THEN 220
210 X = E(I,J)
220 NEXT J
230 NEXT I
240 PRINT "THE LARGEST VALUE 15" X
250 END
```

Result: X = 1510

To find the smallest value, reverse the inequality sign in line 200

Result: X = 356

1A.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 PRINT A(I,J);
50 NEXT J
60 PRINT
70 NEXT I
80 DATA 24,7,8,13,1,19,15,21,2,12
90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
100 END
```

To print B change these lines:

```
30 READ B(I,J)
40 PRINT B (I,J)
80 DATA 1,2,3,4,5,8,9,10,11,12
90 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
```

1B.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR I = 1 TO 5
70 FOR J = 1 TO 5
75 READ B(I,J)
76 NEXT J
77 NEXT I
80 DATA 24,7,8,13,1,19,15,21,2,12
90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
95 DATA 1,2,3,4,5,8,9,10,11,12
97 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
100 FOR I = 1 TO 5
110 FOR J = 1 TO 5
120 IF A(I,J) < 10 THEN PRINT " ";A(I,J);: GOTO 140
130 PRINT A(I,J);
140 NEXT J
145 PRINT TAB(32);
150 FOR K = 1 TO 5
151 IF B(I,K)< 10 THEN PRINT " ";B(I,K);: GOTO 170
160 PRINT B(I,K);
170 NEXT K
180 PRINT
190 NEXT I
200 END
```

Note: Lines 120 and 150 line the numbers up evenly.

```
 24   7   8  13   1          1   2   3   4   5
 19  15  21   2  12          8   9  10  11  12
 16  14   6  10  20          6   7  13  14  15
  4   9   5   3  22         16  17  18  19  25
 25  11  18  23  17         20  21  22  23  24
```

2. Note: Lines 100, 160 provide an additional method of lining numbers up evenly.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR I = 1 TO 5
70 FOR J = 1 TO 5
80 READ B(I,J)
90 NEXT J
95 NEXT I
100 X$="###  ###  ###  ###  ###"
110 DATA 24,7,8,13,1,19,15,21,2,12
115 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
120 DATA 1,2,3,4,5,8,9,10,11,12
125 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
130 FOR I = 1 TO 5
140 FOR J = 1 TO 5
150 C(I,J)=A(I,J)+B(I,J)
160 PRINT USING X$; C(I,J);
170 NEXT J
175 PRINT
180 NEXT I
190 END
```

```
 25    9   11   17    6
 27   24   31   13   24
 22   21   19   24   35
 20   26   23   22   47
 45   32   40   46   41
```

3. Note: Lines 105, 130, and 135 provide another method of lining numbers up evenly.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR I = 1 TO 5
70 FOR J = 1 TO 5
75 READ B(I,J)
76 NEXT J
77 NEXT I
80 DATA 24,7,8,13,1,19,15,21,2,12
90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
95 DATA 1,2,3,4,5,8,9,10,11,12
97 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
100 FOR I = 1 TO 5
105 L=0
110 FOR J = 1 TO 5
120 D(I,J)=A(I,J) -B(I,J)
130 PRINT TAB(L); D(I,J);
135 L = L+5
140 NEXT J
150 PRINT
160 NEXT I
170 END
```

```
 23   5    5    9   -4
 11   6   11   -9    0
 10   7   -7   -4    5
-12  -8  -13  -16   -3
  5 -10   -4    0   -7
```

4.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR I = 1 TO 5
70 FOR J = 1 TO 5
75 READ B(I,J)
76 NEXT J
77 NEXT I
80 DATA 24,7,8,13,1,19,15,21,2,12
90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
95 DATA 1,2,3,4,5,8,9,10,11,12
97 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
100 FOR I = 1 TO 5
105 L = 0
110 FOR J = 1 TO 5
115 FOR K = 1 TO 5
120 E(I,J)=E(I,J) + A(I,K) * B(K,J)
130 NEXT K
135 PRINT TAB(L); E(I,J);
136 L = L + 7
140 NEXT J
150 PRINT
160 NEXT I
170 END
```

```
 356     409      502      555      673
 537     606      780      849      928
 724     790      886      952      1068
 594     637      705      748      806
 929     1023     1207     1301     1510
```

5.

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR J = 1 TO 5
65 K = 0
70 FOR I = 1 TO 5
80 PRINT TAB(K);A(I,J);
85 K = K+5
90 NEXT I
100 PRINT
110 NEXT J
120 DATA 24,7,18,13,1,19,15,21,2,12
125 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
```

```
 24   19    16    4     25
 7    15    14    9     11
 18   21    6     5     18
 13   2     10    3     23
 1    12    20    22    17
```

50

6. Note: To find the smallest value, reverse the inequality sign in line 200. Result: X = 356

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 READ A(I,J)
40 NEXT J
50 NEXT I
60 FOR I = 1 TO 5
70 FOR J = 1 TO 5
75 READ B(I,J)
76 NEXT J
77 NEXT I
80 DATA 24,7,8,13,1,19,15,21,2,12
90 DATA 16,14,6,10,20,4,9,5,3,22,25,11,18,23,17
95 DATA 1,2,3,4,5,8,9,10,11,12
97 DATA 6,7,13,14,15,16,17,18,19,25,20,21,22,23,24
100 FOR I = 1 TO 5
110 FOR J = 1 TO 5
115 FOR K = 1 TO 5
120 E(I,J)=E(I,J) + A(I,K) * B(K,J)
130 NEXT K
140 NEXT J
160 NEXT I
170 X=E(1,1)
180 FOR I = 1 TO 5
190 FOR J = 1 TO 5
200 IF X>=E(I,J) THEN 220
210 X = E(I,J)
220 NEXT J
230 NEXT I
240 PRINT "THE LARGEST VALUE IS " X
250 END
```

```
THE LARGEST VALUE IS  1510
```

Objective: B6a - The student will write a BASIC program to determine the value of a rank-N determinant.

Reference: *Problem Solving With the Computer*, by Edwin Sage, 1969. Entelek, Inc. Pp. 193-209.
Algebra and Trigonometry, Structure and Method, Book 2, Houghton-Mifflin, 1977. Pp. 439-444.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
The student should practice evaluationg 2 x 2, 3 x 3 and larger determinants by the method of introducing zeros in all locations below the main diagonal (triangular reduction). Once he understands the method he can attempt to write a program for a 2 x 2 determinant. This program is easy to generalize to the N x N case. During the triangular reduction method it may happen that it is impossible to continue the reduction process until an interchange of rows or columns of the determinant is performed. Be alert to this problem in case the student encounters it. A general program is given in the Program Appendix, No. B6a.	Supplement B6a

Supplement B6a

EVALUATION OF DETERMINANTS

The triangular reduction method, well-suited to computer evaluation, depends on the fact that the value of a determinant remains unchanged if each element of a row is replaced by the original value plus a fixed multiple of the element in some other row.

Thus
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = \begin{vmatrix} a & b & c \\ d+5a & e+5b & f+5c \\ g & h & i \end{vmatrix}$$

As an example,

$$\begin{vmatrix} -1 & 2 & -3 \\ 2 & 1 & -1 \\ 1 & -1 & 2 \end{vmatrix} = \begin{vmatrix} -1 & 2 & -3 \\ 2+2(-1) & 1+2(2) & -1+2(-3) \\ 1+1(-1) & -1+1(2) & 2+1(-3) \end{vmatrix} = \begin{vmatrix} -1 & 2 & -3 \\ 0 & 5 & -7 \\ 0 & 1 & -1 \end{vmatrix}$$

$$= \begin{vmatrix} -1 & 2 & -3 \\ 0 & 5 & -7 \\ 0-\frac{1}{5}(0) & 1-\frac{1}{5}(5) & -1-\frac{1}{5}(-7) \end{vmatrix} = \begin{vmatrix} -1 & 2 & -3 \\ 0 & 5 & -7 \\ 0 & 0 & 2/5 \end{vmatrix} = (-1)(5)(2/5) = -2$$

The value of the reduced determinant is just the product of the diagonal elements.

The student should practice several such reductions.

The following program uses the above method to evaluate the 3 x 3 determinant given above:

```
 10 DIM A(3,3)
 20 FOR R = 1 TO 3
 30 FOR C = 1 TO 3
 40 READ A(R,C)
 50 NEXT C,R
 60 FOR K = 1 TO 2
 70 FOR R = K TO 3
 80 FOR C = 3 TO K STEP-1
 90 A(R,C) = A(R,C) - A(R,K)/A(K,K)*A(K,C)
100 NEXT C,R,K
110 D = 1
120 FOR R = 1 TO 3
130 D = D*A(R,R)
140 NEXT R
150 PRINT "DET=";D
160 DATA -1,2,-3
165 DATA 2,1,-1
170 DATA 1,-1,2
```

In line 80, the STEP-1 is necessary to avoid changing A(R,K) too soon, since it must remain constant during an entire row reduction. This program is easy to generalize to the NxN case.

Program B6b of the Program Appendix gives the general program, including the case where it may be necessary to interchange rows during the reduction.

MATRIX INVERSION

Let

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 3 & 5 & 7 \end{bmatrix}, \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \text{ and } A = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Then the system of equations can be written as

$MX = A$ (See Suppl. B6.2 - Multiplication of Matrices)

Then $M^{-1}MX = M^{-1}A$, or $X = M^{-1}A$, since $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot X = X$.

This means that if the matrix multiplication $M^{-1}A$ is carried out, the values of x, y, and z will be the matrix elements (numbers) in the first, second, and third row, respectively.

Problem 1 Solve the given system of equations by the method of matrix inversion. Find values for M^{-1}, D, x, y, and z.

Answer: $M = \begin{bmatrix} -4 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -1 \end{bmatrix}$, $D = 1$, $X = 3$, $y = 0$, $z = -1$

Problem 2 Solve the system

$$\begin{aligned} x + 2y + 3z &= 0 \\ 2x + 3y + 5z &= 1 \\ 3x + 5y + 7z &= 2, \end{aligned}$$

Find M^{-1}, D, x, y, and z.

Answer: $D = 0$. Therefore there is no inverse matrix. (The problem has no solution.)

Problem 3 Use Program B6a of the Program Appendix to find the inverse of the matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 4 \end{bmatrix}$$

Answer: $D = 4$, $M^{-1} = \frac{1}{4} \begin{bmatrix} -1 & 4 & 1 & -1 \\ 3 & 0 & 1 & -1 \\ 3 & 4 & -3 & -1 \\ -2 & -4 & 2 & 2 \end{bmatrix}$

Note that small roundoff errors are to be expected. Thus 1 may appear in the program output as 0.99999 and 0 may appear as 3.71428E-07. Students are often confused by such roundoff errors, and may not realize that the actual answers 1 and 0.

A good problem to try a program on is

$$\begin{bmatrix} 5 & -1 & 6 & 2 \\ 3 & -1 & 6 & 2 \\ 35 & 4 & 3 & -3 \\ 18 & 0 & 4 & 1 \end{bmatrix} \qquad \text{(Answer: } = -94)$$

To see why the simple program given in this supplement won't always work it is only necessary to consider the determinant.

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 2 & 4 \end{bmatrix} = - \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}$$, and the program can now be applied to

the second form. Check programs on

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 4 \end{bmatrix} \quad \text{(Answer: } = 4) \text{ and } \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 4 \\ 0 & 4 & 4 & 4 \end{bmatrix} \quad \text{(Answer: } = 16)$$

To see if they handle this problem correctly.

SAMPLE TEST

Name____________________
Period____________________

Test on Topic 13

$R = \begin{bmatrix} 4 & 1 & 0 \\ -3 & 2 & 1 \end{bmatrix}$ $U = \begin{bmatrix} -4 & 1 & 2 \\ 2 & -5 & 0 \end{bmatrix}$ $S = \begin{bmatrix} 0 & 6 \\ -2 & 4 \\ 1 & -3 \end{bmatrix}$

1. (10 Pts.)
Compute R + U

2. (10 Pts.)
Compute U-R

3. (10 Pts.)
Compute R.S

4. (10 Pts.)
Compute S.R

Do not solve, but give the dimensions of the resulting matrix of matrix multiplication is possible. (20 Pts.)

____________________ 5. $x = [1\ 0\ 2]$

$y = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ $x.y = ?$

____________________ 6. R.U
(see above)

____________________ 7. $A = \begin{bmatrix} 2 & 0 & 3 \\ 1 & -1 & 2 \end{bmatrix}$

$B = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 0 & -1 \end{bmatrix}$ $A.B = ?$

____________________ 8. U.S.
(see above)

Assuming the MAT commands are on the system, what would be the output of the following program?

(10 Pts.)

9.
```
10 MAT READ X(3,3), Y(3,3)
20 DATA 1,2,3,4,5,6,7,8,9
30 DATA 9,7,5,3,1,2,4,6,8
40 MAT A = X + Y
50 FOR J = 1 TO 3                          10
60 PRINT A (J,J)                            6
70 NEXT J                                  17
80 END
```

10. Write a program to read these two matrices, to add them (A & B), and print the result (C).

$$A \quad \begin{vmatrix} 6 & 2 \\ 4 & 3 \end{vmatrix} \qquad\qquad B \quad \begin{vmatrix} 0 & 1 \\ -1 & 2 \end{vmatrix}$$

Bonus: Solve any problems (5-8) that can be solved.

ANSWER KEY
SAMPLE TEST

Name________________________________
Period______________________________

Test on Topic 13

$R = \begin{bmatrix} 4 & 1 & 0 \\ -3 & 2 & 1 \end{bmatrix}$ $U = \begin{bmatrix} -4 & 1 & 2 \\ 2 & -5 & 0 \end{bmatrix}$ $S = \begin{bmatrix} 0 & 6 \\ -2 & 4 \\ 1 & -3 \end{bmatrix}$

1. (10 Pts.)
Compute R + U $\begin{bmatrix} 0 & 2 & 2 \\ -1 & -3 & 1 \end{bmatrix}$

2. (10 Pts.)
Compute U-R $\begin{bmatrix} -8 & 0 & 2 \\ 5 & -7 & -1 \end{bmatrix}$

3. (10 Pts.)
Compute R.S $\begin{bmatrix} -2 & 28 \\ -3 & -13 \end{bmatrix}$

4. (10 Pts.)
Compute S.R $\begin{bmatrix} -18 & 12 & 6 \\ -20 & 6 & 4 \\ 13 & -5 & -3 \end{bmatrix}$

Do not solve, but give the dimensions of the resulting matrix of matrix multiplication is possible. (20 Pts.)

____1 X 1____ 5. $x = [1\ 0\ 2]$ $y = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ $x.y = ?$

____No____ 6. R.U (see above)

____2 X 3____ 7. $A = \begin{bmatrix} 2 & 0 & 3 \\ 1 & -1 & 2 \end{bmatrix}$ $B = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 0 & -1 \end{bmatrix}$ $A.B = ?$

____2 X 2____ 8. U.S. (see above)

Assuming the MAT commands are on the system, what would be the output of the following program?

(10 Pts.)

9.
```
10 MAT READ X(3,3), Y(3,3)
20 DATA 1,2,3,4,5,6,7,8,9
30 DATA 9,7,5,3,1,2,4,6,8
40 MAT A = X + Y
50 FOR J = 1 TO 3                        10
60 PRINT A (J,J)                          6
70 NEXT J                                17
80 END
```

10. Write a program to read these two matrices, to add them (A & B), and print the result (C).

A $\begin{bmatrix} 6 & 2 \\ 4 & 3 \end{bmatrix}$ B $\begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$

```
 10 DIM A(2,2),B(2,2),(2,2)
 20 FOR I = 1 TO 2
 30 FOR J = 1 TO 2
 40 READ A(I,J)
 50 NEXT J
 60 NEXT I
 70 FOR I = 1 TO 2
 80 FOR J = 1 TO 2
 90 READ B (I,J)
100 NEXT J
110 NEXT I
120 DATA 6,2,4,3
125 DATA 0,1,-1,2
130 FOR I = 1 TO 2
140 FOR J = 1 TO 2
150 C(I,J) = A (I,J) + B (I,J)
160 PRINT C(I,J);
170 NEXT J
180 PRINT
190 NEXT I
200 END
```

Bonus: Solve any problems (5-8) that can be solved.

5. 5

7. $\begin{bmatrix} 12 & 0 & -1 \\ 7 & -1 & -2 \end{bmatrix}$

8. $\begin{bmatrix} 0 & -26 \\ 10 & -8 \end{bmatrix}$

TOPIC FIFTEEN: CONVERGING PROCESSES

Performance Objectives:

The student will -

B7 Write BASIC programs which use a converging loop and appropriate iterative funtions, to determine the square root of a positive real number and/or the cube root of a positive or negative real number.

B7a To determine all real roots of a quadratic, a cubic, a quartic, or a guintic equation.

Reference

Textbook: pages 431-451

COMPUTER MATHEMATICS

Objective: B7 - The student will write BASIC programs which use converging loop to determine square root of a positive real number and/or the cube root of a positive or negative real number.

Reference: Pages 431-433; 435-438

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Components of converging loops 1. Initial guess(es). 2. Testing Procedure to determine whether the answer or a reasonable approximation has been reached. 3. Formula(s) for producing new guesses. 4. Re-entry into the loop. 5. Exit from the loop.	Textbook Oral Exercise One, p. 433, #1-4 Objective Activity, pp. 445, #1-7 Written Problem Set Fourteen, p. 449, #1-3
Carefully go over the use of testing and formula steps. This includes problems where only an approximation is reached.	Textbook Oral Exercise Two, p. 438, #1-5 Written Problem Set Fourteen, p. 450, #4

COMPUTER MATHEMATICS

Objective: B7 - Convering loops (continued)

Reference: Pp. 441-451

TEACHING SUGGESTIONS

Using converging loops

Cube Root (P. 441)

Square Root (p. 443, #3)

The textbook provides the convergence equations for the cube root example on p. 441, the square root problem (#3, p. 443), and several other problems in the topic. Senior math students should be reminded of Newton's iteration formula,

$$X_2 = X - \frac{f(x)}{f'(x)}$$ where x_2 is the "new guess" x is the "old guess"

This formula is the source of these equations found in Topic 14.

Here is an example:

To find the cube root of N:

let $x = N^{1/3}$

$\therefore x^3 = N$ and $f(x) = x^3 - N$

STUDENT ACTIVITIES

Textbook

Oral

Exercise Three, p. 443, #1-4

Objective Activity, pp. 445- 446, #11-15

Written

Problem Set Fourteen, p. 450, #5, 6

Programs

Problem Set Fourteen, p. 450, #7 - 9

TEACHING SUGGESTIONS	STUDENT ACTIVITIES

Using the formula,

$$x_2 = x - \frac{x^3 - N}{3x^2}$$

$$= \frac{3x^3 - x^3 + N}{3\,x^2}$$

$$= \frac{2\,x^3}{3\,x^2} + \frac{N}{3x^2}$$

$$= \frac{1}{3}\left(2x + \frac{N}{x^2}\right)$$ the formula used in the book

Carefully go over problems given in textbook.

COMPUTER MATHEMATICS

Objective: B7a - The student will write BASIC programs which use a converging loop and appropriate iterative functions to determine all real roots of a quadratic, a cubic, a quartic, or a quintic equation.

Reference: Topic 14, pp. 441-446. Problem 15, p. 446

TEACHING SUGGESTIONS

As discussed for Objective B7, the iterative functions used to solve zeros of polynomials can be found from Newton's Method.

Thus for $f(x) = ax^2+b$, and the iterative function becomes

$x - \frac{f(x)}{f^1(x)} = \frac{ax^2 - c}{2ax + b}$ giving

the repeating formula $x = \frac{9x^2-c}{2ax+b}$

Similarly the iterative formulas for cubic, quartic, and quintic equations are found to be, respectively,

$\frac{2ax^3+bx^2-d}{3ax^2+2bx+c}$, $\frac{3ax^4+2bx^3+5x^2-e}{4x^3+3ax^2+2bx+c}$,

and $\frac{4ax^5+3bx^4+26x^3+dx^2-f}{5ax^4+4bx^3+36x^2+2dx+2}$

Note that the iterative functions give rapid convergence to the correct solutions. This would not be time for the iterative function.

$\frac{2ax^2-c}{3ax+b}$, for example, even though

$x = \frac{2ax^2-c}{3ax+b}$ has the same solutions

as $x = \frac{ax^2-c}{2ax+b}$

STUDENT ACTIVITIES

1. Use the Newton's Method formula

   $x_1 = x_0 - \frac{f(xa)}{f^1(xo)}$ to derive the iterative

   functions appropriate for solving

   a. $ax^2+bx+c = 0$

   b. $ax^3+bx^2+cx+d = 0$

   c. $ax^4+bx^3+cx^2+dx+e = 0$

   d. $ax^5+b+4+cx^3+dx^2+ex+f = 0$

2. Use an iterative loop to solve

   a. $x^2 + x-12 = 0$

   b. $x^3 - 4x^2 -5 x + 20 = 0$

   c. $4x^4 - 8x^3 - 7x^2 - 18x-6 = 0$

   Solutions are:

   a. 3, -4

   b. 4, $\sqrt{5}$, $-\sqrt{5}$

   c. -1.52119, 0.436136,
   1.19832, 1.88673

3. (An extension of the objective for a strong math student.)

   Derive an iterative function to solve
   +anx = x

   Solution: $\frac{x\ seo^2x - xanx}{+an^2x}$

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
	However overflow error occurs unless the initial estimate is close to the correct solutions: 4.49441, 7.72525, 10.9041,... (A better iterative function, converging very rapidly, is $n\pi + \mathrm{Tan}^{-1}x$, where N=1 gives 4.49341, N=2 gives 7.72525,...)

SAMPLE TEST

Quiz on Topic 14

1. Identify the parts of the loop.
(25 Pts.)

Amy has the same number of dimes as quarters. How many of each coin does she have if their total value is $2.80?

```
10 X = 1
20 IF X*.10 + X*.25 = 2.80 THEN 50
30 X = X + 1
40 GO TO 20
50 PRINT X "DIMES AND" X "QUARTERS"
60 END
```

_______________ a. line to which program goes upon exit of the loops

_______________ b. testing procedure

_______________ c. re-entry to loop

_______________ d. initial guess

_______________ e. formula to produce new guess

2. Write a program to find 3 consecutive even integers whose sum is 42.
(25 Pts.)

ANSWER KEY

SAMPLE TEST

Quiz on Topic 14

1. Identify the parts of the loop.
   (25 Pts.)

   Amy has the same number of dimes as quarters. How many of each coin does she have if their total value is $2.80?

```
10 X = 1
20 IF X*.10 + X*.25 = 2.80 THEN 50
30 X = X + 1
40 GO TO 20
50 PRINT X "DIMES AND" X "QUARTERS"
60 END
```

50 a. line to which program goes upon exit of the loops

20 b. testing procedure

40 c. re-entry to loop

10 d. initial guess

30 e. formula to produce new guess

2. Write a program to find 3 consecutive even integers whose sum is 42.
   (25 Pts.)

```
10 X = 2
20 Y = X + 2
30 Z = X + 2
40 IF X + Y + Z = 42 THEN 70
50 X = X + 2
60 GO TO 20
70 PRINT X, Y, Z
80 END
```

ADDITIONAL OBJECTIVES

Among the following topics are several (the "a" objectives) provided for those classes that complete the textbook and have additional time. Others (B2, B3, B4, B10, B14, and B17) are included because there is no specific place in the textbook where those objectives are addressed.

The discussions of both types of objectives contained herein are sequentially grouped according to their objectives number. (The objectives-to-topics cross index can be used to locate the discussion of each objective whether or not it is included in the present list.)

Performance objectives included:

B2, B3 and all objectives from B8a through B17a(j)

COMPUTER MATHEMATICS

Objective: B2 - The student will demonstrate how integers can be stored in memory more compactly than real numbers.

Reference: Curriculum Guide - Objective B4, discussed under Additional Topics B4

TEACHING SUGGESTIONS

Integers, if restricted in range, can be expressed in fewer bytes than can real numbers.

Discuss with the student the binary representation of integers and real numbers. Supplement B2 gives further information on this point.

Students should have mastered the conversion or decimal to binary numbers (Objective B4) before working with the present objective.

STUDENT ACTIVITIES

(See Supplement B2)

1. Represent -7 as a 16 - bit binary number in two's complement invitation.

   Add 16 and -7 in 16 - bit binary notation and obtain the ordinary base 10 representation for their sum.

2. Find the value of NOT 7.

3. Find the value of NOT (NOT 7).

Answers:

1. 1111 1111 1111 1001
2. -8
3. 7
   order problem (optional)
4. Give an 8-bit number to represent the characteristic and a 24-bit number for the mantissa of the computer representation of 123456.75

Answer: 1001 0001
0111 0001 0010 0000 0110 0000

Supplement B2 Computer Representation of Integers and Real Numbers

In a typical microcomputer intergers restricted to the range of values -32768 to +32767 can be stored in two bytes of memory. Numbers outside this range, although perhaps integer in appearance, are considered to be real numbers, and require four bytes of memory for storage. (Some computers treat all numbers as real. Others will treat numbers as integers if they are so designated. By doing so it may be possible to reduce considerably the amount of memory required to store or run program.)

In binary notation the number 32767 is represented as the 16-bit, or 2-byte number (assuming 8 bits per byte, as is time for most microprocessors) 0111 1111 1111 1111, or 2^{15} -1.

Negative numbers can be represented by what is called a "two's complement" notation. In this notation, a negative number N is represented by the number obtained by subtracting 1 from the additive inverse -N, in binary notation, and replacing each 0 by 1 and each 1 by 0. Thus, the representation of -1 is obtained by subtracting 1 from +1, obtaining 0000 0000 0000 0000, and interchanging 0's and 1's, obtaining 1111 1111 1111 1111.

Similarly

1000 0000 0000 0000 represents one less than the negative of 0111 1111 1111 1111, or -(32767)-1 = 32768.

As a simple notation, if the word NOT is used to represent this interchange of 0's and 1's, then the above examples show that

NOT 0 = -1, NOT -1 = 0, NOT 32787 = -32768, and in general, N + NOT N = -1.

Take value of such a notation for negative integers is that computers can subtract by adding.

Consider 5 - 3. In 16-bit notation.

```
         5 = 0000 0000 0000 0101
NOT     -3 = 1111 1111 1111 1101
```

Adding bit by bit, one obtains 1 0000 0000 0000 0010. Finally, dropping the left-hand carry bit (only 16 bits are allowed in the memory bytes), one checking the answer +2 = 0000 0000 0000 0010.

Thus 5 - 3 = 5 + (-3) = +2, obtained by addition and use of the two's complement representation of negative integers.

Real numbers can be represented by using an exponential notation. Thus, 51247.75 can be written as 0.5124775×10^5, The number 5 is called the characteristic and the number 0.5124775 the mantissa. One byte is required to store the characteristic, and three the mantissa. Of course, binary notation is used.

Thus, $51247.75_{10} = 1100\ 1000\ 0010\ 1111.11_2 = 0.1100\ 1000\ 0010\ 1111\ 11_2 \times 2^{16}$, or

in a wholly binary notation, $0.1100\ 1000\ 0010\ 1111\ 11 \times 10^{10000}$.

In this case the characteristic is 10000, or 0001000, requiring one byte of storage. Using one bit for denoting sign and the remaining seven for the value of the number, as described above, exponents as large as $2^7 -1$, or 127 can be represented. Then real numbers up to (but not including) 2^{127}, or approximately 1.7×10^{38}, can be represented. (In practice, a slightly different way of handling a negative characteristic is usually used. It is common to add 1000 0000 to each characteristic, so that 1111 1111 represents a multiplier of 2^{127} and 0000 0001, which is 1000 0000 - 0111 1111, represents 2^{-127}.)

Finally, the mantissa of our problem is 0.1100 1000 0010 1111 11. Since the high order (left hand) bit is always adjusted to be 1, the binary poiant can be used to indicate the sign of the number, as described earlier. Expanding to a full 3 bytes, or 24 bits, the mantissa can now be represented as 0100 1000 0010 1111 1100 0000, where the initial 0 denotes a positive number.

With this notation 23 bits are used to represent significant figures. Since 2^{33} is about $10^{6.9}$, representation of real numbers is accurate to about 7 significant figures.

COMPUTER MATHEMATICS

Objective: B3 - The student will describe possible pitfalls when doing computer arithmetic and methods for avoiding them.

Reference: Textbook - Appendix A - Debugging, p. 451

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Read in Appendix A about rounding error and overflow error.	Study the examples presented in Program B3 of the Program Appendix.
Rounding, or round off, error can become extremely important when two numbers are subtracted which are almost exactly equal in size. In extreme cases the computer result can be wrong by 100% or more.	
Underflow error is similar to overflow error, and occurs when a number is obtained which is so small that the computer cannot distinguish it from zero. (See the discussion of computer representation of numbers, Supplement 8.2.) Most computers will simply replace an "underflowed number" by zero, and continue with the calculatdion in progress.	
Another type of error occurs if one programs with the expectation that numbers such as SQR(25), 2 3, 10 2, or 10*(0.1) are exact integers. On many systems they are not, because of round-off errors in the representation or decimal numbers or in the process of evaluating the indicated functions.	
Program B3 in the Program Appendix examines several examples of errors of the type discussed, and shows steps that can be taken to avoid them. These lessons should be learned well. If ignored, it is quite possible, for example, to include several non-prime numbers in a list of supposed primes (Obj. A17(c), B17a(i))	

COMPUTER MATHEMATICS

Objective: B8a - The student will write BASIC programs using high-resolution graphics and any other special features available on the system.

Reference: A14a - The student will write BASIC programs which utilize SET and RESET.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Use Supplement B8a as an example on graphics. Remember each system is slightly (or very) different.	

Supplement B8a

Graphics for the TRS-80 Microcomputer

Commands -

SET turns on (or lights up) a particular section, block, or "light" on the screen.

RESET turns off (or blackens) a particular "light."

For graphics, the screen is divided into a large number of sections. (Refer to the Video Display Worksheet.) Each "light" is a rectangular block 2 dots wide by 8 dots high; and each has its own "address."

For example:

```
SET (55, 32)   X is the horizontal address
               Y is the vertical address
```

(Do this on the computer - try different addresses.)

Now try RESET (55,32)

Program to create a blinking light:

```
10 CLS*                        *CLS - Clears the screen - it is a good
20 X=60                               habit to use this in all graphics
30 Y=25                               programs.
40 SET (X,Y)
50 RESET (X,Y)
60 GO TO 40
70 END
```

Horizontal Addresses - These are 128 light-block addresses, numbered from 0 to 127 (left to right).

Vertical Addresses - There are 48 light-block addresses, numbered from 0 to 47 (top to bottom).

(If the X or Y value exceeds its limit, an error message appears.)

This program fills the screen from left to right. Redesign it so it starts at the top and fills to the bottom. Next, rewrite it so it starts at the bottom and fills to the top.

```
20 FOR X = 0 TO 127
30 FOR Y = 0 TO 47
40 SET (X,Y)
50 NEXT Y
60 NEXT X
99 GO TO 99
```

Write a program which will allow painting only a small portion of the screen.

This program gives us a straight horizontal line across the entire screen.

```
 10 INPUT "VERTICAL ADDRESS (0 TO 47)"; Y
 20 CLS
 30 FOR X = 0 to 127
 40 SET (X,Y)
 50 NEXT X
 99 GO TO 99  -- to avoid returning control to the prompt - this line locks
                 the computer in an endless loop.  Use the BREAK key to
                 get out of this loop.
```

The CLEAR key resets every block on the screen to off in one operation.

Write a program for a vertical line. Vary the length of the line.

This program draws a rectangle:

```
 10 INPUT "HORIZONTAL STARTING POINT (0 TO 127)"; X
 20 INPUT "VERTICAL STDARTING POINT (0 TO 47)"; Y
 30 INPUT "LENGTH OF EACH SIDE (IN BLOCKS) (0 TO 47)"; K
 40 CLS
 50 FOR L = X, TO Y + K
 60 SET (L,Y)
 70 SET (L,Y + K)
 80 NEXT L
 90 FOR M = Y TO Y + K
100 SET (X,M)
110 SET (X + K, M)
120 NEXT M
999 GO TO 999
```

Broken lines are created by using a step other than 1 or -1. Try this program that creates the illusion of motion.

```
 10 INPUT "VERTICAL ADDRESS (0 TO 47)"; Y
 20 INPUT "STEP SIZE"; S
 30 CLS
 40 FOR X = 0 TO 127 STEP S
 50 SET (X,Y)
 60 NEXT X
 70 Y = Y + 1
 80 GO TO 40
 99 GO TO 99
```

Another program to try:

```
 10 CLS
 20 FOR M = 0 TO 127
 30 SET (M,0)
 40 SET (M, 47)
 50 NEXT M
 60 Y = 2
 70 D = 1
 80 RESET (64, Y-D)
 90 SET (64, Y)
100 Y = Y + D
110 IF Y = 48 THEN 130
120 IF Y < > -1 THEN 80
130 V = Y - 2 * D
140 D = - D
150 GO TO 90
999 GO TO 999
```

COMPUTER MATHEMATICS

Objective: B9a - The student will write BASIC programs using data files, sequential files, and random files.

Reference: Radioshack TRSDOS Manual
Microsoft BASIC - 80 Manual

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
File commands and operations vary greatly from system to system. If disk drives are available, make every effort to expose the students to disk storage of data.* Files are the only way which make data processing feasible for large-scale business and technical applications.	Assignment Supplement B9a.2 Quiz Supplement B9a.3
The size of files may have to be limited because of the lack of disk storage space, but the ideas for creation and manipulation of files remain the same for 20 records or 100 records.	
See Supplement B9a.1	
See also Program B9a of the Program Appendix, which gives examples of files used on a different system (Compucolor II).	

---

*Files can be created on tape, but in most of the equipment available for instruction, it is a time-consuming task with poor results.

Introduction to Files

A file is a logical collection of records. Both programs and data sets are stored as distinct files. Data files are used to handle large amounts of data. Rather than 100 DATA lines in a program, the data can be stored in a separate location of file. A data file is not really a program because it can't be executed; it is used only by being accessed by a program. One data file can be accessed by many program files, and this makes for efficient machine use.

There are two types of data files that can be created and accessed. Sequential files are easy to create, but it is difficult to make additions to them. Random files take up less room on disk and are more flexible.

The following information applies to a system using MICROSOFT BASIC software. TRSDOS is similar with only slight variations in syntax.

Sequential file:
(Steps needed to create a file and access data.)

1) Open in output ("O") mode.
Ex: `OPEN "O", #1, "DATA"`

2) Write data to the file.
Ex: `PRINT #, A$; B$; C$`
(Write #) may also be used.

3) To access data, first close the file.
Ex: `CLOSE #1`
Then reopen in the "I" mode.
Ex: `OPEN "I", #1, "DATA"`

4) Use INPUT # statement tdo read data into a program.
Ex: `INPUT #1, X$, Y$, Z$`

Random File:

1) Open and close in "R" mode.
Ex: `OPEN "R", #1, "DATA"`

2) Describe the buffer field used for input and output.
Ex: `FIELD #1, SAS A$, 10 AS B$`

3) Rename data in program -
LSET (left-justify) and RSET (right-justify) values into the buffer field.
Ex: `LSET X $ = A $`
A $ = name of data in program, and X$ = name of same data in file; these variables must be different.
MKD$, MKI$, and MKS$ must be used to convert numerical data into string data before storing.
Ex: `LSET P$ - MKS$(C)`

4) Transfer data to file.

Ex: `PUT #1, REC %`

REC % stands for record numbers; each record in a random file has a distinct # associated with it - any variable name could be used--examples - C%, N%, CODE %. The % sign stores the variable as an integer, thus taking up less memory space.

If record numbers are not supplied by the programmer, the computer will use consecutive record numbers beginning at 1. If the programmer uses a beginning number of 50, 49 blank records are created, thus wasting storage space on the diskette.

To add to an already existing random file without skipping or writing over records, use

```
LET R = LOF(1) + 1
PUT #1, R
```

(LOF stands for length of file and tells the number of the last record presently used in the file.)

5. Retrieve data from file.

Ex: `GET #1, REC %`

CVD, CVI, CVS must be used to convert string data back into numerical data before displaying.

Ex: `PRINT CVS(A$)`

Sample Programs:

Sequential Files:

1) Program to create a data file:

```
 10 OPEN, "O", #1, "DATA"
 20 INPUT "PROJECT"; P$
 30 IF P$ = "DONE" THEN 80
*40 INPUT "DATE BEGUN"; P$
 50 INPUT "PROJECT LEADER"; L$
 60 PRINT #1, P$; ","; D$; "," L $
 70 PRINT:  GO TO 20
 80 CLOSE 1
 90 END
```

*Data can be a string using hyphens such as 1-12-80

2) Program to access the sequential data file:

```
 10 OPEN "I", #1, "DATA"
*15 IF EOF(1) THEN 50
 20 INPUT #1, P$, D$, L$
 30 IF RIGHT $ (D$,2) = "80" THEN PRINT P$
 40 GO TO 15
 50 CLOSE 1
 60 END
```

*This line stops the program if the end of file is reached during the search for projects begun in 1980.

Random Files:

1) Program to create a data file:

```
 10 OPEN "R", #1, "DATA"
 20 FIELD #1, 20 AS A$, 8 ASB $, 20 AS C$
 30 INPUT "2-DIGIT CODE"; REC %
 40 IF REC % = 999 THEN 130
 50 INPUT "PROJECT"; P $
 60 INPUT "DATE BEGUN"; D $
 70 INPUT "PROJECT LEADER"; L$: PRINT
 80 LSET A$ = P$
 90 LSET C$ = L$
100 LSET C$ = L$
120 GO TO 30
130 CLOSE 1
140 END
```

2) Program to access a data file:

```
 10 OPEN "R", #1, "DATA"
 20 FIELD #1, 20 AS A$, 8 AS B$, 20 AS C$
 30 INPUT "2-DIGIT CODE"; REC %
 40 IF REC % = 999 THEN 100
 50 GET #1, REC %
 60 PRINT A $
 70 PRINT D $
 80 PRINT L $
 90 GO TO 30
100 CLOSE 1
110 END
```

Data file assignment:

1) Set up an external sequential file with the names of 20 students and their test scores. Rewrite the file with the names of the students rearranged in order from the highest scoring to the lowest and print it.

2) Set up a sequential file of names and addresses. Add data to the file and rewrite it.

3) Create a random file (of your choice). Call up the file and print out selected items.

Answers to Supplement B9a.2

```
1)   10 OPEN "O", #1, "LIST"                )
     20 INPUT "NAME"; N$                    )
     30 IF N$ = "DONE" THEN END             )    Sets up file
     40 INPUT "TEST SCORE"; TS              )
     50 PRINT #1, N$; ","; TS               )
     60 PRINT:  GO TO 20                    )
                                        ____
      5 DIM A$(20), G(20)                   )
     10 OPEN "I", #1, "LIST"                )
     20 FOR N = 1 TO 20                     )
     30 INPUT #1, N$, TS                    )    Recalls file, enters it into
     40 LET A$(N) = N$                      )            an array
     50 LET G(N) = TS                       )
     60 NEXT N                              )
     70 CLOSE #1                        ____)  _____
     80 FOR N = 1 TO 19                                )
     90 IF G(N)> = G(N+1) THEN 140                     )
    100 LET Z = G(N)                                   )
    110 LET G(N) = G(N+1)                              )    Sorts list, rearranging
    120 LET G (N+1) = Z                                )    alphabetically, transfers
    125 LET B$ = A$(N)                                 )    scores according to
    126 LET A$(N) = A$(N+1)                            )    new alphabetical list
    127 LET A$(N+1) = B$                               )
    130 GO TO 80                                       )
    140 NEXT N                                         )
    150 OPEN "O", #1, "XLIST"           ____     ______)
    160 FOR N = 1 TO 20                     )
    170 LET NA$ = A$(I)                     )
    180 LET TS = G(I)                       )
    190 PRINT #1, N$; ",";TS                )  Puts new list of scores in new file
    200 PRINT                               )
    210 NEXT END                            )
    220 END                             ____)
```

```
2)    10 OPEN "O", #1, "KK"          )
      20 INPUT N$                    )
      30 IF N$ = "DONE" THEN 70      )
      40 INPUT A                     )     Sets up file
      50 WRITE #1, N$,A              )
      60 GO TO 20                    )
      70 CLOSE #1                    )
      80 END                         )

      10 OPEN "I", #1, "KK"          ________)
      20 OPEN "O", #2, "JJ"                  )
      30 IF EOF(1) THEN 65                   )
      40 INPUT #1, N$,A                      )
      50 PRINT N$,A                          )
      55 WRITE #2, N$,A                      )
      60 GO TO 30                            )
      65 CLOSE #1                            )    Adds data to file;
      70 INPUT N$                            )    rewrites it
      80 IF N$ = "DONE" THEN 110             )
      85 INPUT A                             )
      90 WRITE #2, N$,A                      )
     100 GO TO 70                            )
     110 CLOSE #2                            )
     120 KILL "KK"                           )
     130 NAME "JJ" AS "KK"                   )
     140 END                         ________)

      10 OPEN "I", #2, "KK"      ____)
      20 IF EOF(2) THEN END          )
      30 INPUT #2, N$, A             )    Prints out new file with
      40 PRINT N$, A                 )    added data items
      50 GO TO 20                    )
      60 END                     ____)
```

3) Follow examples in Supplement 9a.1 for random files.

ALTERNATE ANSWERS

```
10 PRINT "1-INITIALIZE FILE"
11 PRINT "2-RECALL FILE, ENTER INTO AN ARRAY"
12 PRINT "3-SORT LIST,TRANSFER NAMES ACCORDING TO GRADES, HIGHEST FIRST"
13 PRINT "4-INITIALIZE NEW FILE WITH SCORES IN ORDER"
14 PRINT "5-PRINT NEW SORTED FILE"
15 INPUT "CHOICE";N
16 ON N GOSUB 100,200,300,400,500
17 GOTO 10
100 OPEN "O",#1,"LIST"
110 INPUT "NAME";N$
120 IF N$="DONE" THEN 170


130 INPUT "TEST SCORE";TS
140 WRITE #1,N$;TS
150 PRINT: GOTO 110
170 CLOSE 1
180 RETURN
200 DIM A$(20),G(20)
210 OPEN "I",#1,"LIST"
220 FOR N = 1 TO 20
230 INPUT #1,N$,TS
240 LET A$(N) = N$
250 LET G(N) = TS
260 NEXT N
270 CLOSE 1
280 RETURN
300 FOR N = 1 TO 19
310 IF G(N)>=G(N+1) THEN 340
320 LET Z=G(N)
330 LET G(N)=G(N+1)
331 LET G(N+1) = Z
332 LET B$=A$(N)
333 LET A$(N)=A$(N+1)
334 LET A$(N+1)=B$
335 GOTO 300
340 NEXT N
350 RETURN
400 OPEN "O",#1,"XLIST"
410 FOR N = 1 TO 20
420 LET N$=A$(N)
430 LET TS=G(N)
440 WRITE #1,N$;TS
450 NEXT N
460 CLOSE 1
470 RETURN
500 OPEN "I",#1,"XLIST"
510 FOR N = 1 TO 20
520 INPUT #1,N$,TS
530 LPRINT N$,TS
540 NEXT N
550 CLOSE 1
560 RETURN
600 END
```

ALTERNATE ANSWERS (continued)

JONES	100
POLL	99
BERKMAN	99
GOW	98
FARBER	97
SHUI	96
PAPE	94
JANSSON	93
FITTER	92
LEE	91
WANG	90
CAUDLE	89
BROD	89
SHOEMAKE	88
BILLINGS	88
MASTERSON	87
WILLIAMS	86
INGRAM	85
NELSON	82
SMITH	78

ALTERNATE ANSWERS (continued)

```
10 PRINT "1-INITIALIZE FILE"
20 PRINT "2-PRINT OUT INITIAL FILE"
30 PRINT "3-ADD DATA TO FILE AND REWRITE IT"
40 PRINT "4- PRINT OUT NEW FILE WITH ADDED DATA ITEMS"
45 PRINT "5-EXIT PROGRAM"
50 INPUT "CHOICE";N
60 ON N GOSUB 100,200,300,400,500
65 GOTO 10
100 OPEN "O",#1,"LIST"
110 INPUT "NAME";N$
120 IF N$ = "DONE" THEN 170
130 INPUT "GRADE";A
140 WRITE #1,N$,A
150 GOTO 110
170 CLOSE 1
180 RETURN
200 OPEN "I",#1,"LIST"
210 IF EOF(1) THEN 260
220 INPUT #1,N$,A
230 PRINT N$,A
240 GOTO 210
260 CLOSE 1
270 RETURN
300 OPEN "I",#1,"LIST"
310 OPEN "O",#2,"XLIST"
320 IF EOF(1) THEN 360
330 INPUT #1,N$,A
340 WRITE #2,N$,A
350 GOTO 320
360 CLOSE 1
370 INPUT "NAME";N$
380 IF N$="DONE" THEN 390
385 INPUT "GRADE";A
387 WRITE #2,N$,A
389 GOTO 370
390 CLOSE #2
395 KILL "LIST"
397 NAME "XLIST" AS "LIST"
399 RETURN
400 OPEN "I",#1,"LIST"
410 IF EOF(1) THEN 460
420 INPUT #1,N$,A
430 PRINT N$,A
440 GOTO 410
460 CLOSE 1
470 RETURN
500 END
```

Name ______________________
Period __________

POSSIBLE TEST

Test 1 - BASIC Files

1. Name two kinds of files in BASIC.

2. Under what mode do you open a sequential file to recall it and bring its data into the CPU?

3. Under what mode do you open a sequential file to initialize it and put data into it?

4. Answer question #2 if it is a random file.

5. Answer question #3 if it is a random file.

6. Print #1 can also be written as

   a. INPUT #1 b. WRITE #1
   c. PRINT 1 d. PUT 1

7. To avoid an INPUT PAST END error message, check for

   a. CLOSE b. LOF
   c. EOF d. FIELD

TRUE OR FALSE:

8. All sequential files require FIELD statements.

9. A data file can be used by only one program.

10. All variables must be entered as strings in a random file.

11. An opening random file # of 120 creates blank records 1 --> 119.

Correct any errors: This program is designed to write the names and scores of students to a file.

```
10 OPEN "I", #1, "DATA"
20 INPUT "NAME"; N
30 IF N = "DONE" THEN END
40 INPUT "SCORE"; S
50 INPUT #1, N; ",; S
60 END
```

Correct any errors in this <u>segment</u> of a program.

```
10 OPEN "R", #2, "FILE"
20 FIELD #1, 20 AS N$, 4 AS A$
30 INPUT "2-DIGIT CODE"; CODE %
40 INPUT "NAME"; N$
50 LSET N$ = X$
```

Name____________________________
Period____________

ANSWER KEY

POSSIBLE TEST

Test 1 - BASIC Files

1.	Name two kinds of files in BASIC.	random sequential
2.	Under what mode do you open a sequential file to recall it and bring its data into the CPU?	I
3.	Under what mode do you open a sequential file to initialize it and put data into it?	O
4.	Answer question #2 if it is a random file.	R
5.	Answer question #3 if it is a random file.	R
6.	Print #1 can also be written as a. INPUT #1 b. WRITE #1 c. PRINT 1 d. PUT 1	b
7.	To avoid an INPUT PAST END error message, check for a. CLOSE b. LOF c. EOF d. FIELD	c

TRUE OR FALSE:

8.	All sequential files require FIELD statements.	F
9.	A data file can be used by only one program.	F
10.	All variables must be entered as strings in a random file.	T
11.	An opening random file # of 120 creates blank records 1 --> 119.	T

Correct any errors: This program is designed to write the names and scores of students to a file.

```
           O
10 OPEN "I", #1, "DATA"
20 INPUT "NAME"; N$
30 IF N$= "DONE" THEN END
40 INPUT "SCORE"; S
   PRINT
50 INPUT #1, N; ",; S
60 END   55 GO TO 20
```

Correct any errors in this segment of a program.

```
10 OPEN "R", #2, "FILE"
             2
20 FIELD #1, 20 AS N$, 4 AS A$
30 INPUT "2-DIGIT CODE"; CODE %
40 INPUT "NAME"; N$ X$
50 LSET N$ = X$
```

COMPUTER MATHEMATICS

Objective: B10 - The student will write and run programs with business applications or other applications appropriate to student interests.
B17a (h) Produce a payroll, an inventory, or solve some banking problem.

Reference:

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Objective B10	Samples
Discuss the use of computer programs in many fields.	Supplement B10
Then have students prepare programs applicable to their fields of interest.	
Objective B17a(h)	
Program 3 of Supplement B10 solves a banking problem. It shows how to program an automatic teller computer to print checks in a given format.	

Sample Business Application Programs

1. A businessman own shares in five different companies. The first line of data contains the number of shares owned in each company. The second line of data contains the current values of these five stocks, respectively. Find the total value of his stock shares.

   DATA 50, 210, 540, 76, 82
   DATA 25, 55.20, 112.25, 33.50, 45

2. Read the following data into an array. Create another array with last names first and alphabetize it. Use it to alphabetize the first array and then print the first array, alphabetized.

   DATA "GEORGE COOK", "JAMES SMITH", "BOB DALY", "HOWARD MILLER", "TOM REED"

   (lack of string space may limit the list of names)

3. Program an automatic teller computer to print checks in the format below. (The symbol b̸ denotes a blank.)

   Skip 9 lines

   b̸b̸b̸ PAY b̸ TO b̸ THE b̸ ORDER b̸ OF b̸ J. b̸ P. b̸ JONES

   Skip 1 line

   ******* $12,346.30 b̸ DOLLARS

   Skip 8 lines

   ↓

   123 b̸ b̸ 456 b̸ b̸ 7 <-- Account # beginning in 20th space
   Skip 1 line
   DATA "JOHN PAUL JONES", 1234567, 12345.6
   DATA "ABRAHAM GETTYSBURG LINCOLN", 9876543, 104384.72
   DATA "RUDOLPH KLOTZ GUGGENHEIMER", 7654321, 84.8

   Print all checks on the same prize, write VOID on them immediately.

#2

```
10 INPUT "HOW MANY NAMES ARE ON THE LIST";N
20 FOR I = 1 TO N
30 READ X$(I)
40 NEXT I
50 FOR I = 1 TO N
60 FOR K = 1 TO LEN(X$(I))
70 IF MID$(X$(I),K,1)=" " THEN 90
80 NEXT K
90 T = LEN(X$(I))-K
100 Y$=RIGHT$(X$(I),T)
110 Z$(I)=Y$ + " " + LEFT$(X$(I),K-1)
120 NEXT I
130 DATA "GEORGE CLARK","JAMES SMITH", "BOB DALY", "HOWARD MILLER"
135 DATA "TOM REED"
140 FOR I = 1 TO N-1
150 IF Z$(I)<=Z$(I+1) THEN 210
160 A$=Z$(I)
170 B$=X$(I)
180 Z$(I)=Z$(I+1)
190 X$(I)=X$(I+1)
195 Z$(I+1)=A$
200 X$(I+1)=B$
205 GOTO 140
210 NEXT I
220 FOR  I = 1 TO N
230 PRINT X$(I)
240 NEXT I
250 END
```

```
GEORGE CLARK
BOB DALY
HOWARD MILLER
TOM REED
JAMES SMITH
```

#3

```
10 DATA "JOHN PAUL JONES", 1234567,12345.6
20 DATA "ABRAHAM GETTYSBURG LINCOLN",9876543,104384
30 DATA "RUDOLPH KLOTZ GUGGENHEIMER", 7654321, 84.8
40 READ N$,X$,Y
50 FOR I = 1 TO 9
60 PRINT " "
70 NEXT I
80 PRINT TAB(4); "PAY TO THE ORDER OF ";
90 PRINT LEFT$(N$,1);".";
100 FOR I = 1 TO LEN(N$)
110 IF MID$(N$,I,1)=" " THEN 130
120 NEXT I
130 K = LEN(N$)-I
140 M$=RIGHT$(N$,K)
150 P$=LEFT$(M$,1)
160 PRINT P$;".";
170 FOR J = 1 TO LEN(M$)
180 IF MID$(M$,J,1) = " " THEN 200
190 NEXT J
200 K = LEN(M$)-J
210 Q$=RIGHT$(M$,K)
220 PRINT Q$
230 PRINT
240 A$="**$#,###,###.## DOLLARS"
250 PRINT USING A$;Y
260 FOR I = 1 TO 8
270 PRINT " "
280 NEXT I
290 C$ = LEFT$(X$,3)
300 D$=MID$(X$,4,3)
310 F$=RIGHT$(X$,1)
320 PRINT TAB(20);C$," ";D$;" ";F$
330 PRINT " "
340 GOTO 40
350 END
```

```
    PAY TO THE ORDER OF J.P.JONES
*****$12,345.60 DOLLARS




                 123        456 7




    PAY TO THE ORDER OF A.G.LINCOLN
****$104,384.00 DOLLARS



                 987        654 3




    PAY TO THE ORDER OF R.K.GUGGENHEIMER
*********$84.80 DOLLARS



                 765        432 1
```

```
1.   5 PRINT "THE TOTAL VALUE OF STOCK IS";
    10 FOR I = 1 TO 5
    20 READ N(I)
    30 NEXT I
    40 FOR I = 1 TO 5
    50 READ M(I)
    60 NEXT I
    65 S = 0
    70 FOR I = 1 TO 5
    80 S = S + M(I)*N(I)
    90 NEXT I
   100 A $ = "$$#####.##"
   110 PRINT USING A$;S
   120 DATA 50, 210, 540, 76, 82
   130 DATA 25, 55.20, 112.25, 33.50, 45
   140 END

OUTPUT:  THE TOTAL VALUE OF STOCKS IS $79693.00
```

COMPUTER MATHEMATICS

Objective: B11a: The student will write a simple assembly language program.

Reference: None

Other Reference: CARDIAC - A Cardboard Illustrative Aid to Computation, Bell Telephone Laboratories (1968)

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Most computers, from micro on up, can be programmed in an assembly language which is basically machine language written in a mnemonic code. Common microcomputer assembly languages are based on a variety of microcomputers. The Z-80 (TRS-80, Sinclair), 8080 (Compucolor, Sol, Heath), and 6502 (Pet, Atari, Appale) are perhaps the most common. The simplified assembly language developed by David Hagelbarger and Saul Fingerman and explained in the booklet CARDIAC, is easy to learn and apply. A limited supply of these books may still be available free of charge from Southwestern Bell. Contact the Staff Supervisor for Educational Relations. In Houston, this is Billie Marable, who can be reached at 521-8530.	Supplement B11a

Supplement B11a

CARDIAC Assembly Language

CARDIAC uses the following assembly language mnemonics*:

0 INP
1 CLA
2 ADD
3 TAC
4 SFT
5 OUT
6 STO
7 SUB
8 JMP
9 HRS

Variables can be stored in any of 99 addresses, ranging from 01 to 99. (00 is a ROM location in which is always stored instruction or number 001.) Thus if variable A is stored in address 32, the machine language instruction 132 becomes the assembly instruction CLA A. The instruction tells the CARDIAC computer to clear the acculumator and enter into it the valve of the contents of address 32.

A brief explanation of the 10 assembly codes follows:

032 - INPA - input, or read a data card, entering its numerical value into memory cell 32, the address of variable A.
132 - CLAA - clear accumulator and enter the value of A into the accumulator.
232 - ADD A - add the value of A to the present accumulator contents.
345 - TAC45 - test accumulator. If its value is negative, jump to (take the next instruction form) address 45. Otherwise the next instruction is the one immediately following the present one.
432 - SFT32 - shift the accumulator contents 3 spaces to the left, then 2 spaces to the right. The accumulator holds only 4 digits, and those shifted off to the left are simply dropped, then replaced by zeros during the right portion of the shift. Thus if the accumulator contents were 5678 the instruction SFT32 would result in contents 0080.
532 - OUT A - print the value of A.
632 - OUT A - store accumulator contents in the address of variable A.
732 - SUB A - subtract the value of A from the present accumulator contents.
845 - JMP45 - jump, or transfer program control, to the instruction in location 45.
945 - HRS45 - halt program operation and reset the pointer (which marks the address of the next program instruction) to address 45.

---

* Copyright 1968, Bell Telephone Laboratories. Reprinted by permission.

Sample Programs:

I. Variables

Address	Name
35	B

Constant

Address	Contents
00	001

A. Machine Language

Address	Contents
21	100
22	635
23	535
24	200
25	822

B. Assembly Language

Address	Contents
21	CLA 1
22	STO B
23	OUT B
24	ADD 1
25	JMP 22

C. BASIC Language

Line No.	Statement
10	B = 1
20	PRINT B
30	B = B + 1
40	GO TO 20

Note that there is not a one-to-one correspondence between machine or assembly language and BASIC statements. The correspondence can be made closer if the symbol X is used to represent accumulator contents. Then the BASIC program could be expanded to

```
10 X = 1
20 B = X
30 PRINT B
40 X = X + 1
50 GO TO 20
```

II. Variables

Address	Name
36	A

Constants

Address	Contents
0D	001
01	-004

A. Machine Language

Address	Contents
20	101
21	200
22	636
23	536
24	321
25	900

B. Assembly Language

Address	Contents
20	CLA-4
21	ADD 1
22	STO A
23	OUT A
24	TAC 21
25	HRS 00

C. BASIC Language

```
10 A = - 4
20 A = A + 1
30 PRINT A
40 IF A < 0 THEN 20
50 END
```

III. Variables

Address	Name
50	A
51	B
52	C

Input Values

Card	Value
1	13
2	7

A. Machine Language

Address	Contents	Address	Contents
19	051	24	650
20	052	25	550
21	152	26	401
22	410	27	650
23	751	28	550
		29	900

B. Assembly Language

```
INPB        STO B
INPC        OUT A
CLA C       SFT 01
SUB 8       OVT A
            HRS 00
```

C. BASIC Language

```
10 READ B
20 READ C
30 DATA 13,7
40 A = 10*C-B
50 PRINT A
60 A = INT(A/10)
70 PRINT A
80 END
```

The CARDIAC booklet gives several examples of use of machine language. The reader is left to construct his own assembly and BASIC programs.

The logic of computer operation is briefly explained in introductory chapters. In later chapters the reader is shown how to bootstrap a program into the computer, so that by beginning in address 00 with the 001 (INP 1) instruction, the entire program is loaded into CARDIAC and then executed. It is also shown how to store the address of the exit point from calling program so that re-entry return can be made from a subroutine.

Working with the CARDIAC booklet should serve not only an introduction to assembly language programming but should also develop in the student a deeper understanding of the logic of BASIC language programming. Even a teacher wishing, ultimately, to teach assembly language programming specific to the computer at his school, should consider introducing the fundamental ideas with the help of the simplified CARDIAC language.

COMPUTER MATHEMATICS

Objective: B12a, B12b - The student will write programs involving at least three of the following: simulation, probability, decision trees, linear and/or binary searches, game theory, computer assisted instruction, Boolean algebra and logical operators, cryptanalysis, three-dimensional representations and manipulations, pattern recognition. Expand one of the topics into a term project.

References:

Books:

David Ahl, editor, BASIC Computer Games, Microcomputer Edition, Workman Publishing, 1978.

David Ahl, editor, More BASIC Computer Games, Creative Computing Press, 1979.

William Ralph Bennett, Jr., Introduction to Computer Applications for Non-Science Students (BASIC), Prentice-Hall, Inc., 1976.

Thomas Dwyer and Margot Critchfield, BASIC and the Personal Computer, Addison-Wesley Publishing Company, Inc., 1978.

A.I. Forsythe, Keenan, Organick and Stenberg, Computer Science, A First Course, John Wiley & Sons, 1969.

C.W. Gear, Introduction to Computer Science, Short Edition, SRA, Inc., 1973.

Fred Gruenberger and George Jaffray, Problems for Computer Solution, John Wiley & Sons, 1965.

Vern Court Hare, Jr., Introduction to Programming: A BASIC Approach, Harcourt, Brace & World, Inc., 1970.

Herbert Maisel, Computers, Programming and Applications, SRA, Inc., 1976.

Forrest M. Mims, III, Understanding Digital Computers, Radio Shack, 1978.

James Poirot and David Groves, Computers and Mathematics, Sterling Swift Publishing Company, 1979.

Stephen J. Rogowski, Problems for Computer Solution, Teacher Edition, Creative Computing Press, 1980.

Donald D. Spencer, Problems for Computer Solution, 2nd ed., Hayden Book Company, Inc., 1977.

Magazines, Specific References

BYTE, November, 1980, High Resolution Graphics issue.

Joel C. Hungerford, "Graphic Manipulations Using Matrices," BYTE, September, 1978, pp. 156-165.

Jeffrey L. Posdamer, "The Mathematics of Computer Graphics," BYTE, September, 1978, pp. 22-39.

General Magazine References

BYTE; Creative Computing; Kilobaud Microcomputing - monthly issues

Since the subject material of these objectives is not present in the textbook, these subjects should be regarded as a list of research or mini-research topics for the student. For the aid of the student (and teacher), references are presented in this discussion for each subject mentioned. Probably no single reference and quite possibly no list of two or even three references would contain discussion of each of the subjects mentioned. The sixteen specific references mentioned in this section, taken together, discuss them rather thoroughly, however, and a few minutes of library work should turn up other relevant books or articles. The three journals mentioned, each issued monthly, are important sources of information about the above and similar computer-related topics.

The subjects will briefly be referenced one at a time.

Simulation - a numerical technique for conducting experiments on a computer.

Best reference -	Dwyer and Critchfield, Chapter 9
Others -	Gruenberger & Jaffray, Problems E4 and E5
	Van Court Hare, pp. 319-328

Several examples are given by David Ahl (first book) as the games Chemist, Hammurabi, Life, Lunar Lem Rocket, Splat, Civil War, King.

Probability - references are not really needed, as materials is readily available. Rogowski has 7 problems, Spencer has 90, Gurenberger and Jaffray have several, for example.

Decision Trees - A decision tree is a tree used to make a decision. A bit hard to track down, such trees are discussed by Dwyer and Critchfield, page 60 and applied to calorie counting and by Forsythe et al, pp. 373-375, who consider the problem of a minimal number of weighings.

As to the question of what a tree itself is, this information is easier to locate, and is important, at an advanced level, to the process of data manipulation. Trees are discussed by Bennett (p. 108), Gear (Sect. 7.3, pp. 281-296) and in considerable detail by Forsythe et al (Chapters 9 and 11).

Linear and/or binary searches - The basic idea is simply illustrated. You are trying to guess, successively, 0, 1, 2, 3, 4, ..., until you have the right one. Number of guesses? About 50, on the average, using this linear search method. If you can determine whether your guess is too high or too low, however, you can perform a binary search by guessing in the middle of the unknown interval. After each guess the unknown interval is halved. Thus the guessing might go: 50, 25, 37, 31, 34, 32, 33. Seven guesses instead of 50.

Linear and binary searching are discussed by Gear, pp. 197-199.

Binary searching is also discussed by Van Court Hare, pp. 269-271.

Game Theory - This topic could be interpreted to mean learning something of the mathematical discipline founded by Von Neumann and discussed in numerous mathematical treatised, or simply, learning how to write BASIC programs which are game simulations. Both topics are discussed by Dwyer and Critchfield in Chapter 6. Their book, again, is best reference to this Topic A. A poorer discussion can be found in Forsythe et al, pp. 130-132, and of course, there are numerous examples of programmed games in the Ahl books.

Computer-Assisted Instruction - Using a computer to teach. A sufficient example occurs in the present curriculum guide. See the discussion of Objective B14a. Ahl has several CAI or CAI-like pograms in his books, including Animal, Literature Quiz, Train, and Eliza.

Boolean Algebra and Logical Operations - The logic of the logical operators OR, NOT, AND is sometimes called Boolean algebra. This logic is applied in electronic theory to the construction of logic circuitry which is used in computers to odd numbers. A discussion of this sort of application is given by Mims in Chapter 3 of his book.

Cryptanalysis - Study of codes, or secret writing. Books on the subject abound. Among those written from a compluter point of view are those by Van Court Hare, pages 303-308, and Bennett. The latter treatment is quite detailed, covering 93 pages (Chapter 4). Incidentally, as mentioned in the notes for Program B4 in the program Appendix, conversion from one number base to another can be used as a coding device. In Ahl's first book, the games Bagels and Mastermind deal with cryptanalysis of sorts.

Three-Dimensional Representations and Manipulations - Supplement B5 in the Topic 12 discussion shows one approach to graphing three-dimensional function: letting numbers code the size of the function. Such a graph is essentially a contour map of a function and resembles a map of elevation contours such as is produced in a geological survey. Another approach, taken by Bennett, Chapter 3, pages 86-101, is to attempt to produce actual contour lines in a three-dimensional perspective, to give a sense of depth to the computer output. It is also possible to represent solid objects in perspective, and determine how they look from different angles. This is the approach of Hungerford and of Posdamer in their BYTE magazine articles. The latter approach can become rather involved mathematically, incidentally.

Pattern Recognition - A simple interpretation of this toic is that of locating particular patterns or groupings of characters in a given text. This is a relatively easy problem involving string functions. For example, locate the substring "BUG" in the string "BABBY BUGGY." Some BASICs, in fact, have an INSTR function which is designed precisely to make such an identification. This type of problem is discussed by Forsythe et al, pages 474-487.

What is actually meant by pattern recognition in the present case, however, is the ability of a computer to recognize geometric shapes, as, for example, letters or numbers. Such recognition is basic to the porblem of "teaching a computer to read." This, more interesting problem, is discussed by Bennett in Chapter 2, pages 67-72. The treatment is somewhat, perhaps necessarily, mathematical.

COMPUTER MATHEMATICS

Objective: B13a: The student will write BASIC programs using the PEEK function and the POKE command.

Reference:

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
The POKE. I, J statement is used to place a number J, say between 0 and 255, into memory location I. The PEEK (I) function is used to learn what number already is stored in memory location I. Description and knowledge of the system used are required in using PEEK and POKE. Poking a number into the wrong memory location could result in having to reload BASIC, for example, or in switching output from one device to another. On the other hand, POKE-ing and PEEK-ing at the proper locations can give the programmer more control over operation of his computer system than he would otherwise have. The particular memory locations to be used and the role they play, unfortunately, vary from computer to computer; the best advice is: Consult your programming manual! Program B13a of the Programming Appendix shows a program using PEEK and POKE to access the Compucolor II real time clock.	

COMPUTER MATHEMATICS

Objective: B14: The student will write BASIC programs using the string functions LEFT$, RIGHT$, MID$, STR$, LEN, VAL, ASC, AND CHR$.

Reference: BASIC and the Personal Computer, by Thomas Dwyer and Margot Critchfield, Addison-Wesley Publishing Company (1979), Chapter 4

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Have the students study the list of string functions in Supplement 14. Give blackboard or overhead projector illustrations of the usage. (See Program B14 of the Program Appendix which gives a number of examples.)	Learn the names and use of the string functions described in Supplement B14.1.
Have students practice using each of these string functions by writing several short programs, each containing at least three of the functions.	
Have students check their knowledge or string function by working the problems in Supplement B14.2.	Supplement B14.2.

Ten String Functions

(Definitions are valid for Compucolor II. Minor variations may occur for other systems.)

A$ stands for any legitimate string variable.

1. ASC(A$) - gives ASCII code for the 1st character only of A$, or zero for the null string (an 8-bit code). [Null string is "".]

2. CHR$(N) - for 0≤N≤255 gives ASCII character number N. If N>127 the same as CHR$(N-128).

3. FRE(X$) - (here X$ doesn't stand for a previously-defined string) - gives the number of free string bytes if you say PRINT FRE(X$). More can be cleared with the CLEAR N command. (For completeness note that PRINT FRE(X) gives the number of unused bytes in RAM.)

4. LEN(A$) - gives number of characters in A$, including blanks.

5. LEFT$(A$,I) - gives the I left-most characters of A$, or all if I≥LEN(AS). Thus LEFT$("APPLE",3) equals "APP".

6. RIGHT$(A$,I) - gives the I right-most characters of A$, or all if I≥LEN(A$). Thus RIGHT$("APPLE",3) equals "PLE".

7. MID$(A$,I) - gives all characters of A$, beginning with and to the right of the I th. Gives the null string if I>LEN(A$). Thus MID$(A$,I) is equivalent to RIGHT$(A$,LEN(A$)-I+1).

8. MID$(A$,I,J) - gives J consecutive characters of A$, beginning with the I th. Thus MID$("CREATE",3,2) is the string "EA". 0≤I≤255, 0≤J≤255.

9. STR$(X) - turns the numerical variable X into a string variable. Thus if X has the value 6, RIGHT$(STR$(X+100),2) is the string "06". Invaluable for printing leading zeros, as in "THE TIME IS 2:09:06"

10. VAL(AS$) - gives the value represented by the "string number" A$. Thus VAL("07.13") is 7.13. If A$ doesn't represent a number then VAL(A$) is zero. Note that VAL(STR$(X) is X but that STR$(VAL(AS)) equals A$ only if A$ is the string form of a number.

Supplement B14.2

String Function Exercise

ASCII code (partial list)

Character * + - . / 0 1...9 = ? A B ... J K ... P ... U ... Z

Code 42 43 45 46 47 48 49 57 61 63 65 66 74 75 80 85 90

1. What is the computer output for each of the following?

```
a) PRINT ASC("J")
b) A$="A": PRINT ASC(A$)
c) PRINT CHR$(80)
d) PRINT ASC(J)
e) PRINT CHR$("J")
f) J=63: PRINT CHR$(J)
g) PRINT CHR$(50); CHR$(43); CHR$(01); CHR$(52); CHR$(63)
h) PRINT CHR$(ASC("APPLE"))
i) PRINT ASC(CHR$(75))
```

2. Given A$ = "BANANACREAMPIE", B$ = "DATECAKE"; C$ = "BOOKSTORE", and D$ = "12345", determine each of the following:

```
a) LEFT$(C$,4)
b) RIGHT$(C$,4)
c) MID$(C$,4)
d) MID$(C$,4,4)
e) MID$(B$,2,4)
f) MID$(B$,4,2)
g) LEFT$(A$,6)+MID$(B8,1,4)+RIGHT$(C$,4)
h) LEN(C$)
i) VAL(A$)
j) 2*VAL(RIGHT$(D$,2))
k) LEN(STR$(7))
```

3. Write a program to input a string, then print it out reversed letter for letter. For example, if you input MALE then the program produces the output ELAM.

ANSWER KEY - String Functions Exercise

1. a) 74
   b) 65
   c) P
   d) error
   e) error
   f) ?
   g) 2+3 = 4?
   h) A
   i) 75

2. a) BOOK
   b) TORE
   c) KSTORE
   d) KSTO
   e) ATEC
   f) EC
   g) BANANADATETORE
   h) 9
   i) 0
   j) 90
   k) 2

3.

```
10  INPUT A$
20  B=LEN(A$)
30  FOR I = B TO 1 STEP -1
40  PRINT MID$(A$,I,1);
50  NEXT J
```

or

```
10  INPUT A$
20  FOR I = 1 TO LEN (A$)
30  C$ = MID$(A$,I,1)+C$
40  NEXT I
50  PRINT C$
```

COMPUTER MATHEMATICS

Objective: B14a: The student will write a program which will teach and/or test a short self-contained topic in any field.

B4: The student will convert numbers from base 10 to a smaller base.

Reference:

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
This objective is important in helping students see yet other uses of computers. Supplement 14a.1 is a program teaching the remainder method for base conversion. This is a program that can be done by most students.	Write a teaching and or testing program.

Supplement B14a

Program to Teach the Remainder Method of Base Conversion

Notes

1) This program first tests to see whether the student can convert from base ten to other bases less than ten. If the student scores less than 70% on the test, he is told to see his teacher to review that material. If he scores 70% or better, he is told to continue with the lesson.

2) The student is then taught the "short-cut" method of converting by using the remainder method. Each step is explained.

3) At the end of the lesson a quiz is given and graded. The student is told to show it to his teacher.

4) Two RUNS are included to give examples of tests and lessons completed by two types of students:

    a) a "goof-off" student

    b) a good student

PROGRAM TO TEACH REMAINDER METHOD OF BASE CONVERSION

```
 10 PRINT "HELLO, I'M TERI.  WHAT'S YOUR NAME?;
 20 INPUT N$
 30 PRINT N$;", TODAY WE ARE GOING TO LEARN A NEW METHOD OF -
 40 PRINT "CHANGING FROM BASE TEN TO SMALLER BASES."
 50 PRINT
 60 PRINT "FIRST, LETS BE SURE WE REMEMBER THE OLD METHOD."
 70 PRINT
 80 PRINT "CHANGE THE FOLLOWING FIVE EXAMPLES ACCORDING "
 90 PRINT "TO THE DIRECTIONS."
100 Z=0
110 FOR L=1 TO 5
120 DATA 35,4,61,2,78,6,98,8,49,3,234,5,345,7,456,9,51,2,96,4
130 DATA 109,6,124,8,86,3,147,5,156,7,345,9
140 READ N,P
150 PRINT N;" BASE TEN = ? BASE ";P
160 GOSUB 1320
170 INPUT T
180 IF T=B THEN
190 PRINT " TRY AGAIN ";$;".  WORK MORE CAREFULLY"
200 INPUT T
210 IF T=B THEN 260
220 PRINT "THAT'S STILL NOT QUITE CORRECT"
230 IF L=5 THEN 280
240 PRINT "LETS TRY THE NEXT ONE."
250 GOTO 280
260 PRINT "GOOD.  THAT'S CORRECT."
270 Z=Z+20
280 NEXT L
290 PRINT
300 PRINT "YOUR SCORE IS ";Z;"%."
310 IF Z>=60 THEN 360
320 PRINT "YOU NEED TO WORK A LITTLE MORE ON THE PREVIOUS LESSON."
330 PRINT "ASK YOUR TEACHER TO HELP YOU."
340 END
350 PRINT
360 PRINT "YOU ARE READY FOR THE NEW LESSON."
370 PRINT
380 FOR I=1 TO 5
390 READ N,P
400 A=N
410 B=0
420 C=0
430 PRINT "TO CHANGE ";N;"
440 PRINT "DIVIDE ";N; BY ";P;"."
450 GOSUB 920
460 PRINT "WHAT IS YOUR REMAINDER";
470 INPUT R
480 IF R=F THEN 500
490 GOTO 530
500 PRINT "GOOD.  THIS IS THE LAST DIGIT OF YOUR NEW NUMERAL."
510 PRINT
```

```
520 GOTO 550
530 PRINT "TRY AGAIN. WORK CAREFULLY."
540 GOTO 460
550 PRINT "WHAT IS YOUR QUOTIENT";
560 INPUT Q
570 IF Q=N THEN 590
580 GOTO 610
590 IF N<=0 THEN 650
600 GOTO 630
610 PRINT "TRY AGAIN, ";N$
620 GO TO 550
630 GOSUB 990
640 PRINT
650 PRINT "SINCE YOUR LAST QUOTIENT WAS ZERO, WE NO LONGER"
660 PRINT "HAVE ANYTHING TO DIVIDE."
670 PRINT
680 PRINT A;" BASE 10 = ";B; BASE ";P
690 PRINT
700 PRINT
710 NEXT I
720 PRINT "NOW FOR A REVIEW QUIZ . DO THESE FIVE BY YOURSELF"
730 PRINT "USING THE NEW METHOD,"
740 K=0
750 FOR J=1 TO 5
760 READ N,P
770 PRINT N;" BASE 10 = ? BASE ";P
780 INPUT S
790 GOSUB 1320
800 IF S=B THEN 280
810 GOTO 860
820 K=K+20
830 PRINT "CORRECT"
840 PRINT
850 GOTO 880
860 PRINT "THAT ONE IS NOT CORRECT.  LETS TRY THE NEXT ONE."
870 PRINT
880 NEXT J
890 PRINT "YOUR SCORE IS ";K;"%"
900 PRINT "SHOW YOUR PAPER TO YOUR TEACHER."
910 END
920 FOR F=0 TO (P-1)
930 IF N-INT(N/P)*P=F THEN 950
940 NEXT F
950 B=B+AF*10^C
960 C=C+1
970 N=INT(N/P)
980 RETURN
990 FOR F=0 TO (P-1)
1000 IF N-INT(N/P)*P=F THEN 1020
1010 NEXT F
1020 B=B+F*10^C
1030 PRINT
```

```
1040 PRINT "NOW LETS FIND ANOTHER DIGIT."
1050 PRINT "DIVIDE ";N;" BY ";P
1060 PRINT "WHAT IS YOUR REMAINDER";
1070 INPUT R
1080 IF R=F THEN 1110
1090 PRINT "CHECK YOUR DIVISION AGAIN, ";N$
1100 GOTO 1060
1110 PRINT "GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL."
1120 PRINT "YOU ALREADY HAD."
1130 PRINT "NOW WHAT IS YOUR NEW NUMERAL";
1140 INPUT S
1150 IF S=B THEN 1170
1160 GOTO 1200
1170 C=C+1
1180 N=INT(N/P)
1190 GOTO 1220
1200 PRINT "TRY AGAIN.  THE NEW DIGIT GOES TO THE LEFT."
1210 GOTO 1130
1220 PRINT "WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT";
1230 INPUT Q
1240 IF Q=N THEN 1270
1250 GOTO 1290
1260 PRINT "GOOD."
1270 IF N<=0 THEN 1310
1280 GOTO 990
1290 PRINT "TRY AGAIN.  WORK CAREFULLY."
1300 GOTO 1220
1310 RETURN
1320 B=0
1330 C=0
1340 FOR F=0 TO (P-1)
1350 IF N-INT (N/P)*P=F THEN 1370
1360 NEXT F
1370 B=B*10^C
1380 C=C+1
1390 N=INT (N/P)
1400 IF N>0 THEN 1340
1410 RETURN
```

"GOOF-OFF" RUN - This student obviously does not have any desire to do the assigned work.

```
 RUN
HELLO.  I'M TERI.  WHAT'S YOUR NAME? WHO KNOWS
WHO KNOWS, TODAY WE ARE GOING TO LEARN A NEW METHOD OF
CHANGING FROM BASE TEN TO SMALLER BASES.

FIRST, LETS BE SURE WE REMEMBER THE OLD METHOD.

CHANGE THE FOLLOWING FIVE EXAMPLES ACCORDING
TO THE DIRECTIONS.
 35    BASE TEN = ? BASE 4
?DONT KNOW
INVALID DATUM
?302
 TRY AGAIN WHO KNOWS.  WORK MORE CAREFULLY
?456
THAT'S STILL NOT QUITE CORRECT
LETS TRY THE NEXT ONE.
 78    BASE TEN = ? BASE  2
?32456
 TRY AGAIN WHO KNOWS.  WORK MORE CAREFULLY
?23
THAT'S STILL NOT QUITE CORRECT.
LETS TRY THE NEXT ONE.
 78    BASE TEN = ? BASE   3
?7654
 TRY AGAIN WHO KNOWS.  WORK MORE CAREFULLY
7432
THAT'S STILL NOT QUITE CORRECT.

YOUR SCORE IS 0 %.
YOU NEED TO WORK A LITTLE MORE ON THE PREVIOUS LESSON.
ASK YOUR TEACHER TO HELP YOU.
```

Good Student   RUN

```
 RUN
HELLO.  I'M TERI.  WHAT'S YOUR NAME?  CAROLYN
CAROLYN.  TODAY WE ARE GOING TO LEARN A NEW METHOD OF
CHANGING FROM BASE TEN TO SMALLER BASES.

FIRST, LETS BE SURE WE REMEMBER THE OLD METHOD.

CHANGE THE FOLLOWING FIVE EXAMPLES ACCORDING
TO THE DIRECTIONS.
 35    BASE TEN = ?  BASE   4
?203
GOOD.  THAT'S CORRECT.
 61    BASE TEN = ?  BASE   2
?111101
GOOD.  THAT'S CORRECT.
 78    BASE TEN = ?  BASE   6
7210
GOOD.  THAT'S CORRECT.
 98    BASE TEN = ?  BASE   3
?1811
GOOD.  THAT'S CORRECT.

YOUR SCORE IS 100 %.
YOU ARE READY FOR THE NEW LESSON.

TO CHANGE 234 TO BASE 5 ,
DIVIDE   234  BY  5
WHAT IS YOUR REMAINDER?  4
GOOD.  THIS IS THE LAST DIGIT OF YOUR NEW NUMERAL.

WHAT IS YOUR QUOTIENT?   46

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  46  BY  5
WHAT IS YOUR REMAINDER?  9_1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  14
WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT?  9

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  9  BY  5
WHAT IS YOUR REMAINDER?  4
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  414
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  1

NOW LETS FIND ANOTHER DIGIT.

DIVIDE 1  BY  5
WHAT IS YOUR REMAINDER? 1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  1414
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?   0
```

```
SINCE YOUR LAST QUOTIENT WAS ZERO, WE NO LONGER
HAVE ANYTHING TO DIVIDE.

234    BASE 10 =  1414   BASE 5

TO CHANGE  345 TO BASE  7,
DIVIDE  345  BY  7  .
GOOD.  THIS IS THE LAST DIGIT OF YOUR NEW NUMERAL.

WHAT IS YOUR QUOTIENT?  49

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  49  BY  7
WHAT IS YOUR REMAINDER?  0
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  02
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  7  BY  7
WHAT IS YOUR REMAINDER?  0
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  002
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  1

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  1  BY  7
WHAT IS YOUR REMAINDER?   1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  1002
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  0

SINCE YOUR LAST QUOTIENT WAS ZERO, WE NO LONGER
HAVE ANYTHING TO DIVIDE.

 345   BASE 10 =  1002  BASE 7

TO CHANGE 456 TO BASE 9 ,
DIVIDE 456  BY  9  .
WHAT IS YOUR REMAINDER?  5_6
GOOD.  THIS IS THE LAST DIGIT OF YOUR NUMERAL.

WHAT IS YOUR QUOTIENT?  50

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  50  BY  9
WHAT IS YOUR REMAINDER?  5
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  56
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  5

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  5  BY  9
WHAT IS YOUR REMAINDER?  5
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  556
WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT?  0
```

```
SINCE YOUR LAST QUOTIENT WAS ZERO, WE NO LONGER
HAVE ANYTHING TO DIVIDE.

456    BASE 10 = 556    BASE 9

TO CHANGE  51  TO  BASE  2,
DIVIDE  51  BY  2 .
WHAT IS YOUR REMAINDER?  1
GOOD.  THIS IS THE LAST DIGIT OF YOUR NEW NUMERAL.

WHAT IS YOUR QUOTIENT?  25

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  25  BY  2
WHAT IS YOUR REMAINDER?  1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  11
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  12

NOW LETS FIND ANOTHER DIGIT.
DIVIDE 12  BY  2
WHAT IS YOUR REMAINDER?  0
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  011
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  6

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  6  BY  2
WHAT IS YOUR REMAINDER?  0
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  0011
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  3

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  3  BY  2
WHAT IS YOUR REMAINDER?  1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  10011
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  1

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  1  BY  2
WHAT IS YOUR REMAINDER?  1
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  1100____110011
WHEN YOU DIVIDED, WHAT WAS YOUR QUOTIENT?  0

SINCE YOUR LAST QUOTIENT WAS ZERO.  WE NO LONGER
HAVE ANYTHING TO DIVIDE.

51     BASE 10 =  110011    BASE 2
```

```
TO CHANGE  96  TO BASE  4,
DIVIDE  96  BY  4  .
WHAT IS YOUR REMAINDER?  2
TRY AGAIN.  WORK CAREFULLY.
WHAT IS YOUR REMAINDER?  0
GOOD.  THIS IS THE LAST DIGIT OF YOUR NEW NUMERAL.

WHAT IS YOUR QUOTIENT?  24

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  24  BY  4
WHAT IS YOUR REMAINDER?  0
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  00
WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT?  6

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  6  BY  4
WHAT IS YOUR REMAINDER?   2
 GOOD.  PUT THIS DIGIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
NOW WHAT IS YOUR NEW NUMERAL?  200
WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT?  1

NOW LETS FIND ANOTHER DIGIT.
DIVIDE  1  BY  4
WHAT IS YOUR REMAINDER?  1
 GOOD.  PUT THIS DITIT TO THE LEFT OF THE NUMERAL
YOU ALREADY HAD.
 NOW WHAT IS YOUR NEW NUMERAL?  1200
WHEN YOU DIVIDED , WHAT WAS YOUR QUOTIENT?  0

SINCE YOUR LAST QUOTIENT WAS ZERO, WE NO LONGER
HAVE ANYTHING TO DIVIDE.

 96    BASE  10 =  1200    BASE 4

NOW FOR A REVIEW QUIZ.  DO THESE FIVE BY YOURSELF
USING THE NEW METHOD.
 109   BASE 10 = ?  BASE  6
?130
TEHAT ONE IS NOT CORRECT.  LETS TRY THE NEXT ONE.

 124   BASE 10 = ?  BASE  8
?174
CORRECT

 86    BASE 10 = ?  BASE 3
?174
CORRECT

 147   BASE 10 = ?  BASE 5
?1042
CORRECT

 156   BASE 10 =  ?  BASE 7
?312
CORRECT

YOUR SCORE IS  80%
SHOW YOUR PAPER TO YOUR TEACHER.
```

Objective: B15a: The student will write programs using other programming
language: FORTRAN

Reference: Textbook - Minitopic - Another Language, p. 330

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Teaching FORTRAN -	
If there is FORTRAN available on the system, an introduction to programming in this language may be included in the course. Most textbooks assume no prior programming knowledge when teaching a computer language. After students have learned BASIC, however, another language can be learned more quickly. A good reference to use in comparing FORTRAN with BASIC is found in Computers and Mathematics by Jonas Poirot and David Groves, pp. 199-214. One way to make students feel comfortable with FORTRAN is to take problems previously done in BASIC and translate them into FORTRAN.	Supplement B15a.3
A major difference between FORTRAN and BASIC is the importance of spaces in the lines of a FORTRAN program. Be sure that students are familiar with the four fields and their respective columns: 1) statement label (or number) field, (columns 1-5) 2) continuation character field (column 6) 3) statement field (columns 7-72) 4) identification field (columns 73-80)	Supplement B15a.1

TEACHING SUGGESTIONS

Comment lines (similar to REM statements in BASIC) must have a C in column 1. Continuation lines must have something other than a blank or a zero in column 6. Before students become too involved with fields and formatting, they may be encouraged to write some format-free programs, even if they will not run on the available equipment. Supplement B15a.1 has some examples of this type of problem. The relational expressions should already have been covered (.LT., .LE.,.EQ.,.NE.,.GT.,.GE.) along with the IF...GOTO statement (IF(K.LT.N) GO TO 3).

Valid variables, arithmetic statements, real and integer variables and constants, and the different modes all are topics which need to be covered. A good idea is to keep formatting as simple as possible; establish a general format for both integer variables and real variables. Students may have to be reminded which letters indicate integer variables; the key is in the word integer, I-N. Supplement B15a.2 presents sample exercises to familiarize students with elementary programming in FORTRAN. The IF statments and DO loops will allow more complicated programming. Supplement B15a.3 gives some programming examples.

A comparison of a BASIC subroutine and a FORTRAN subroutine can be seen in Programs B17a(g).2 and B17a(g).3 in the Program Appendix.

STUDENT ACTIVITIES

FORTRAN EXERCISES - I

1. Write a program to read two numbers and subtract the second number from the first.

2. Write a program to add four numbers without using a loop.

3. Redo Number 2, using a loop.

4. Write a program that finds and prints the gross pay for employees knowing the pay rate per hour and the number of hours worked per week. (The number of employees is known.)

ANSWER KEY TO SUPPLEMENT B15a.1

Note: The PRINT and READ statements here are in a computer language called format - free FORTRAN. Not all systems have the capability but it is an easy way to start.

```
     Col.

1.     C    THIS PROGRAM SUBTRACTS 2 NUMBERS
            S = 0
            READ, A, B
            S = A - B
            PRINT, S
            STOP
            END


2.     C    THIS PROGRAM ADDS 4 NUMBERS
            S = 0
            READ, A,B,C,D
            S=A+B+C+D
            PRINT,S
            STOP
            END

         5  7
3.     C    THIS PROGRAM ADDS 4 NUMBERS USING A LOOP
            S = 0
            N = 0
          1 READ,A
            N = N+1
            S=S+A
            IF(N.LT.4) GO TO 1
            PRINT, S
            STOP
            END


4.     C    EMPLOYEES' SALARIES
            N = 0
            READ, K
          1 READ R, H
            W = R * H
            PRINT, W
            N=N+1
            IF (N.LT.K) GO TO 1
            STOP
            END
```

Supplement B15a.2

FORTRAN TEST

A. Each of the following is an invalid FORTRAN variable. Identify the error in each.

1. Y.O.U.
2. G-4
3. 2ND
4. TAX NO
5. SUPPLEMENT

B. Write the following expressions in FORTRAN:

1. $T = 2Y$
2. $P = D^2$
3. $Y = X^3 - B$
4. $(X/Y)^{m-1}$
5. $A(B+1)$

C. Circle the real constants:

375   37.5   100   100.   -8953   0

D. Circle the integer variables:

XNUM   NUM   AREA   WALT   PERIM   KOUNT

E. Identify the mode of the following expressions as integer, real, or mixed:

1. I/J + MEMBER
2. (A+B)*(A-B)
3. PI*AR**2*LENGT
4. ALL + TOTAL - DEBT
5. JOEP/43.9-KYZ

F. Evaluate the following arithmetic statements:

1. M=14.0/2.0+3.0
2. Z=(5**2-10)./10
3. X=3/7
4. NO = 4 + (4+7)/12
5. I = 2 + 3 * 5

G. In the following problems, determine the statement number of the next statement to be executed if Y=4.0 and Z=-3.0.

1. If(Y)10,20,30
2. IF(Z)15,25,35
3. IF(Y+Z)45,5,20
4. W=3.0*Y+4.0*Z
   IF (W)27,28,29

H. What will be printed upon execution of the following segment of a program?

```
      DO 200 I = 1,5,1
      J =2*1
      WRITE (1,1)J
    1 FORMAT (I7)
  200 CONTINUE
```

ANSWER KEY TO SUPPLEMENT B15a.2

A. 1. no periods (no symbols)
2. no dash
3. can't begin with a number
4. no space
5. too long (maximum is 5 or 6 letters, depending on system)

B. 1. T = 2*Y
2. P = D**2
3. Y = X**3-B
4. (X/Y)**(N-1)
5. A*(B+1)

C. 37.5, 100.

D. NUM, KOUNT

E. 1. Integer
2. Real
3. Mixed
4. Real
5. Mixed

F. 1. 10
2. 1.0
3. 0.
4. 4
5. 17

G. 1. 30
2. 15
3. 20
4. 28

H. 2
4
6
8
10

Supplement B15a.3

FORTRAN EXERCISES - II

1. Take the odd integers from 1 to 20, print the number, its square, and its cube, printing the values in columns with appropriate headings.

2. Write a program which will calculate n! for n = 1 to 15 and print out n and n! in appropriately-headed columns.

3. Write a program that will generate the prime #'s less than 300 and list across the page.

4. a. Write a sort program that will select and print out the largest # in a set of data.

   b. Sort all the numbers.

5. Generate n terms of the Fibonacci series $n^{c}20$ and find their sum.

6. Illustrate

   a. using input from the terminal
   b. using a data file
   c. using literals

Sample programs for Supplement B15a.3
(These programs run on a system with MICROSOFT FORTRAN-80.)

#1

```
NUMBER           SQUARE          CUBE

   1                1               1

   3                9              27

   5               25             125

   7               49             343

   9               81             729

  11              121            1331

  13              169            2197

  15              225            3375

  17              289            4913

  19              361            6859
```

```
C     ODD INTEGERS,SQUARES,CUBES
      INTEGER N,S,C
      N=1
      WRITE(2,4)
    4 FORMAT(7H1NUMBER,10X,6HSQUARE,8X,4HCUBE)
    5 S=N**2
      C=N**3
      WRITE(2,6),N,S,C
    6 FORMAT(1H0,I4,10X,I4,10X,I4)
      N=N+2
      IF(N.LT.20) GO TO 5
      STOP
      END
```

#2

```
        NUMBER               N-FACTORIAL
            1.0                       1.0
            2.0                       2.0
            3.0                       6.0
            4.0                      24.0
            5.0                     120.0
            6.0                     720.0
            7.0                    5040.0
            8.0                   40320.0
            9.0                  362880.0
           10.0                 3628800.0
           11.0                39916800.0
           12.0               479001600.0
           13.0              6227020800.0
           14.0             87178289152.0
           15.0           1307674279936.0
```

```
C    THIS PROGRAM CALCULATES N FACTORIAL FOR N=1
C    TO 15 AND PRINTS N AND N! IN APPROPRIATELY
C    LABELED COLUMNS
     REAL N,S
     N=1.
     S=1.
     WRITE(2,1)
   1 FORMAT(1H1,8X,6HNUMBER,15X,11HN-FACTORIAL)
   3 S=S*N
     WRITE(2,2),N,S
   2 FORMAT(F16.1,10X,F16.1)
     IF(N.GE.15) GO TO 4
     N=N+1
     GO TO 3
   4 STOP
     END
```

#3

```
C       PRIME NUMBERS
        DIMENSION NN(100)
        M=0
        WRITE(2,6)
      6 FORMAT(1X,'PRIME NUMBERS')
        DO 3 I = 2,300
        IF(I-3) 5,5,2
      2 K = I/2
        IF(I-K*2.EQ.0) GO TO 3
        X=I
        L=SQRT(X)+1
        DO 4 N=3,L,2
        J = I/N
        IF(I-J*N.EQ.0) GO TO 3
      4 CONTINUE
      5 M = M+1
        NN(M) = I
      3 CONTINUE
      9 WRITE(2,7) (NN(KL),KL=1,M)
      7 FORMAT(12(/1X,6I11))
        STOP
        END
```

```
 PRIME NUMBERS

          2          3          5          7         11         13
         17         19         23         29         31         37
         41         43         47         53         59         61
         67         71         73         79         83         89
         97        101        103        107        109        113
        127        131        137        139        149        151
        157        163        167        173        179        181
        191        193        197        199        211        223
        227        229        233        239        241        251
        257        263        269        271        277        281
        283        293
```

BEST COPY AVAILABLE

#4

```
C     SORT
      DIMENSION N(10)
      DATA N/38,12,0,-5,77,10,9,7,-70,-12/
      WRITE(2,3) (N(L),L=1,10)
    3 FORMAT(//,' THE UNSORTED LIST',/10I5)
      DO 2 I = 1,9
      KK = I + 1
      DO 1 J = KK,10
      IF(N(I).GE.N(J))GO TO 1
      K = N(J)
      N(J)=N(I)
      N(I)=K
    1 CONTINUE
    2 CONTINUE
      WRITE(2,4) (N(L),L=1,10)
    4 FORMAT(//' THE SORTED LIST',/10I5)
      STOP
      END
```

```
1


 THE UNSORTED LIST
   38   12    0   -5   77   10    9    7  -70  -12


 THE SORTED LIST
   77   38   12   10    9    7    0   -5  -12  -70
```

#5

```
C     FIBONACCI SERIES
      DIMENSION N(20)
      NSUM = 0
      N(1) = 1
      N(2) = 1
      DO 1 I = 3,20
      N(I) = N(I-1) + N(I-2)
    1 CONTINUE
      DO 2 I = 1,20
      NSUM = NSUM + N(I)
    2 CONTINUE
      WRITE(2,3) (N(L),L=1,20)
    3 FORMAT(1X,//' THE FIBONACCI SERIES',5(/,10I5))
      WRITE(2,4) NSUM
    4 FORMAT(1X/' THE SUM OF THE SERIES IS', I8)
      STOP
      END
```

```
THE FIBONACCI SERIES
    1    1    2    3    5    8   13   21   34   55
   89  144  233  377  610  987 1597 2584 4181 6765

THE SUM OF THE SERIES IS    17710
```

6a

```
C      USING DATA FROM THE TERMINAL
       INTEGER ANS
    10 WRITE(1,1)
     1 FORMAT(1X,' WHAT IS THE NUMBER? ')
     2 FORMAT(I3)
       READ(3,2) N
C      3 IN THE READ STATEMENT REPRESENTS THE LUN OF THE CRT
       K = N*2
C      K DOUBLES THE NUMBER INPUT
       WRITE(1,3) K
     3 FORMAT(1X,I7)
       WRITE(1,6)
     6 FORMAT(1X,' DO YOU WISH TO INPUT ANOTHER NUMBER? -
      1 ENTER 1 FOR YES AND 2 FOR NO ')
       READ(3,4) ANS
     4 FORMAT(I2)
       IF(ANS.EQ.1) GOTO 10
       STOP
       END
```

6b

```
C      USING A DATA FILE
       CALL OPEN(6,'DATATSSSDAT',0)
     2 READ(6,1) R
       PI = 3.1416
       AREA = PI*R**2
       WRITE(2,1) AREA
     1 FORMAT(F16.4)
       N=N+1
       IF(N.LT.5) GO TO 2
       STOP
       END
```

```
         3.1416
        12.5664
        28.2744
        50.2656
        78.5400
```

6c

```
C      USING LITERALS
       DOUBLE PRECISION A
       DATA A/'JOHN DOE'/
       WRITE(2,1) A
     1 FORMAT(1X,'THE NAME OF THE STUDENT IS ',A8)
       STOP
       END
```

```
THE NAME OF THE STUDENT IS JOHN DOE
```

Objective: B15a: The student will write programs using other programming languages: COBOL

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
Teaching COBOL: If COBOL is available, a brief introduction to the language may be presented in the course. Having previously studied BASIC and perhaps FORTRAN, students will in al probability object to the rigid, elaborate format of a COBOL program. Stress the readability and predictability of a COBOL program and COBOL's use as a language to process files. The four divisions of a COBOL program are easy to remember if students learn this saying, "In Every Darned Program..."; each beginning letter represents one of the four divisions in the correct order. (Identification, Environment, Data, Procedure). A problem which exists with teaching COBOL is the length of the programs. Blackboards and overheads fill too quickly with even a simple COBOL program. Without the use of a textbook, handouts of COBOL programs may prove to be the only other means of illustration aside from actual programs in the computer. Exercises in "outlining" the data division may prove helpful (see #10, Supplement B15a.4) only a limited number of COBOL programs may be written because of the time involved in keeping and compiling. The flexibility of COBOL arithmetic statements should be stressed (see #12, Supplement B15a.4). Supplement B15a.4 is a sample test covering elementary information to be learned about COBOL.	Supplement B15a.4

COBOL EXERCISES

1. Every COBOL program is composed of 4 divisions written in a specific order. They are:

   a. ____________________________

   b. ____________________________

   c. ____________________________

   d. ____________________________

2. What does the acronym COBOL stand for? ____

3. Which of the following Identification Divisions are free of error? (Yes or No - Circle errors)

   ____________ a. IDENTIFICATION-DIVISION.
   PROGRAM-ID. THIS-IS-IT.
   ____________ b. IDENTIFICATION DIVISION
   PROGRAM ID. CLOSING.
   ____________ c. IDENTIFICATION DIVISION.
   PROGRAM-ID. THIS-IS-IT?
   ____________ d. IDENTIFICATION DIVISION.
   PROGRAM-ID. SMITH 01.

4. Fill in the blanks:

   a. A margin begins in column ________.

   b. B margin begins in column ________.

   c. Columns ignored by the computer are ____________.

   d. Statment #'s occupy the first ________ spaces

   e. A hyphen in column ______ indicates a continuation line.

5. The last statement executed in a COBOL program is ____________________________.

6. To get the word "hello" to appear on the screen, the instruction line would be____________________________.

7. To input a value for X from the CRT, the instruction line would be

   ____________________________.

8. What does FD stand for? ____

9. What special lead # do variables in working storage take? ___________

10. Write a record description entry for the following record. Assign an appropriate name to the record.

NAME		PHONE	ADDRESS	TODAY'S DATE			BLANKS
INITIALS	LAST NAME			MONTH	DAY	YEAR	
1	3 29	30 37	38 59	60	62	64	66 80

11. Examples of Data editing (use b̸ to represent planks).

	Source Picture	Source Value	Editing Picture	Edited Item
a)	9999	0012	ZZZ9	
b)	9V9	5‸7	9.9	
c)	999	001	99999	
d)	99	76	9B9	
e)	9999999	0004987	Z,ZZZ,ZZZ	
f)	9V999	6‸124	.999	
g)	S99	+35	99	

12. Which of these decision statements are free of error? (Yes or no - circle errors)

____________ a) IF A EQUALS B SUBTRACT 1 FROM D.

____________ b) IF ALPHA >B THEN STOP RUN.

____________ c) IF A <B * 2 COMPUTE D=A↑2.

13. In COBOL, all numeric data is ______________ justified, and all alphameric data is ______________ justified.

14. Which division in a COBOL program contains PICTURE clauses? ______________

15. Write a COBOL statement to find A if A = $6B+C^2$.

____

16. The COBOL verb used as an equivalent to the DO instruction in FORTRAN is

______________________.

ANSWER KEY TO SUPPLEMENT B15a.4

1. a) Identification
   b) Environment
   c) Data
   d) Procedure

2. Common Business Oriented Language

3. a. NO No dash between IDENTIFICATION and DIVISION.
   b. NO Dash missing between PROGRAM and ID.
   c. NO ? can't be used in a program name.
   d. YES

4. a. 8
   b. 12
   c. 73-80
   d. 6
   e. 7

5. STOP RUN

6. DISPLAY 'HELLO!

7. ACCEPT X

8. file description

9. 77

10.
```
   01      CUSTOMER CARD
           05  CUSTOMER-NAME
               08 INITIAL          PICTURE X(2).
               08 LAST-NAME        PICTURE X(27).
           05 PHONE                PICTURE X(8).
           05 ADDRESS              PICTURE X(22).
           05 TODAYS-DATE
              08 MONTH-NO          PICTURE 99.
              08 DAY-NO            PICTURE 99.
              08 YEAR-NO           PICTURE 99.
           05 FILLER               PICTURE X(15).
```

11. a) 12
    b) 5.7
    c) 00001
    d) 7 ƀ 6
    e) 4, 987
    f) .124
    g) 35

ANSWER KEY to Supplement B15a.4 (continued)

12. a. No E Q U A LⓈ <--- omit

b. Yes

c. No A ↑ 2 <--- ** instead

13. right
left

14. DATA

15. COMPUTE A = 6 * B + C * * 2

16. PERFORM

COMPUTER MATHEMATICS

Objective: B16a* - The student will explain data processing and programming to persons not registered in Computer Mathematics.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
As a culmination of the course, students may be asked to put themselves in the roles of teachers and systems analysts. One of the major problems in the growth of data processing is the lack of computer literacy among the general populace. Communication between data processors and the users of the computer systems is often poor and limits the expansion of data services. Students can observe this phenomenon first hand and perhaps expand computer literacy in their own school by means of a two-fold project. Ask students to find a willing and interested teacher, administrator, clerk, or student group for whom they might perform a data processing service. Examples of services are computerizing textbook records and keeping statistics for the basketball team. The programming groups will need to establish good communication with their user in order to provide a useful service. The project should consist of two parts: (1) a general lesson or lessons for the user covering basic computer terms and elementary programming concepts, and (2) the writing and explaining of the program or series of programs to perform the designated service. (Documentation is very important.) This project will provide students insight into the method and problems of professional data processors.	

COMPUTER MATHEMATICS

Objective: B17(a): The student will find the sum of two fractions, express the sum as a fraction, reduce the sum to the lowest terms, and print out the fractions and their sum.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
It is necessary to find the greatest common factor of the denominators in order to reduce the final fraction to lowest terms. Although the g.c.f. of two numbers can be found in a relatively straighforward manner, the best procedure is to use the Euclidean algorithm. (See Supplement A17a(a).) Programs A17a(a) in the Program Appendix illustrate both a standard but inefficient program and the Euclidean algorithm method for finding the g.c.f. A program for solving the present objective is given as Supplement B17(a).	

Objective: B17(b) - The student will solve, when coupled with a subroutine evaluating a rank-N determinant, a system of four linear equations.

B17(c) - The student will solve, when coupled with a subroutine evaluating the inverse of a matrix, a system of four linear equations.

Reference: Algebra and Trigonometry, Structure and Method, Book 2, Houghton-Mifflin (1977). Chapters about determinants and matrices, pp. 439-457

TEACHING SUGGESTIONS

Objective B17(b):

The discussion of Objective B6b in Topic 13 gives a simple program for a 3 x 3 determinant. The general NXN program is given as Program B6b in the Program Appendix. The method of Cramer's rule for solving a system of linear equations is well known. For the present case, let the coefficient determinant be

$$D = \begin{vmatrix} a & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 4 \end{vmatrix} = \det M$$

Then

$$X = \begin{vmatrix} 4 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 8 & 1 & 0 & 1 \\ 1 & 0 & 3 & 4 \end{vmatrix}, \quad y = \begin{vmatrix} 0 & 4 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 8 & 0 & 1 \\ 1 & 1 & 3 & 4 \end{vmatrix}$$

$$Z = \begin{vmatrix} 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 8 & 1 \\ 1 & 0 & 1 & 4 \end{vmatrix}, \quad w = \begin{vmatrix} 0 & 1 & 1 & 4 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 8 \\ 1 & 0 & 3 & 1 \end{vmatrix}$$

STUDENT ACTIVITIES

Solve for x, y, z and w:

$$\begin{aligned} y + z + w &= 4 \\ x + z + w &= 0 \\ x + y + w &= 8 \\ x + 3z + 4w &= 1 \end{aligned}$$

a) Use Cramer's rule

b) Use matrix methods (matrix inverse and matrix multiplication)

Answer:

$x = 3/4$

$y = 19/4$

$z = 13/4$

$w = 5/2$

TEACHING SUGGESTIONS	STUDENT ACTIVITIES

Objective B17(c):

Program B6a in the Program Appendix inverts an N x N non-singular (det $\neq 0$) matrix, that is, finds the inverse M^{-1} of the coefficient Matrix M. As discussed in Supplement B6a, the matrix product

$$(M^{-1}) \begin{matrix} 4 \\ 0 \\ 8 \\ 1 \end{matrix} \quad \text{equals} \quad \begin{matrix} x \\ y \\ z \\ w \end{matrix} \text{, from which values}$$

can be obtained directly for x,y,z and w.

Actually, Program B6a also solves for x,y,z and w. However, this feature can be suppressed by the following changes:

a) omit line 1913
b) replace line 1950 by
   `FOR J = 1 TO N : FOR I = 1 TON`
c) replace line 1980 by
   `FOR I = 1 ton : FOR J = 1 TON`

COMPUTER MATHEMATICS

Objective: B17a(a) - The student will find the sum or difference of two fractions input in normal algebraic form; answer reduced to lowest terms.

TEACHING SUGGESTIONS

This objective differs from B17(a) in that an entire expression such as 2/3-3/7 is entered, rather than simply the numbers 2, 3, -3, and 7. The computer must then scan the expression, identify each number separately, and note whether a sum or a difference is called for. It does this by inputting the expression as a string variable (See Objective B14) and examining the string character by character to pick off the relevant information.

A student attempting to write such a program should be urged first, to write down his logic in the form of a flowchart, and second, to run a trace (Objective A9) on the flowchart before writing the actual BASIC program.

The student should know the Euclidean algorithm as a means of reducing the final fraction to longest terms. (See Objective A17a(a).)

A BASIC program which illustrates the present objective is Program B17a(a) in the Program Appendix.

STUDENT ACTIVITIES

1. Write a flowchart for a program meeting the project objective.

2. Trace the flowchart for assumed input such as 2/3 - 3/7.

3. Find the greatest common factor of 15 and 35 by using the Euclidean algorithm.

4. Prepare the BASIC program, working from the flowchart, and using the Euclidean algorithm for g.c.f. to reduce the final fraction to lowest terms.

COMPUTER MATHEMATICS

Objective: B17a(b) - The student will find one real and two complex cube roots of a real number.

TEACHING SUGGESTIONS

The algebra of complex numbers tells us that every number, whether real or complex, has two square roots, three cube roots, four fourth roots, etc.

In the case of cube roots of a real number, one must necessarily be real (the principal root) and the other two complex conjugates of each other.

In terms of the representation $\cos(x) + i\ \underline{\sin}(x) = \mathrm{cis}(x) = e^{ix}$, where $i = \sqrt{-1}$, two complex roots are obtained from the real root by multiplying it by

$$e^{\frac{2}{3}\pi in}$$

where n = 1 or 2. Since the computer cannot find the 1/3 power of a negative number, the three desired roots of N are written as

$$\mathrm{SGN(N)*(ABS(N))}^{1/3}$$

$$\mathrm{SGN(N)*(ABS(N))}^{1/3}*[\cos\frac{2\pi}{3} + i\sin\frac{2\pi}{3}]$$

and

$$\mathrm{SGN(N)*(ABS(N))}^{1/3}*[\cos\frac{2\pi}{3} - i\sin\frac{2\pi}{3}],$$

where all angles are given in radions. These results can be simplified, since

$$\cos\frac{2\pi}{3} = -1/2 \text{ and } \sin\frac{2\pi}{3} = \frac{2\pi}{3} = \frac{\sqrt{3}}{2}$$

Program B17a(b) of the Program Appendix lists the desired program.

STUDENT ACTIVITIES

COMPUTER MATHEMATICS

Objective: B17a(c) - The student will add, subtract, or multiply two real numbers of up to 30 digits.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES

With this objective, one enters the domain of what is called multiple-precision arithmetic. A computer program may let us determine that 20! (twenty factorial) has the approximate value 2.43290×10^{18}, but how can one learn, short of reading this curriculum guide, that the exact result is 2432902008176640000? The answer lies in a program for multiple precision multiplication.

Even if a computer system's BASIC interpreter or compiler does not implement double precision arithmetic it is possible for a programmer to write his own routines. What is required is a certain amount of programming skill and a knowledge of the addition, subtraction and multiplication algorithms that are taught in elementary school.

Supplement B17a(c) gives suggestions for approaches to the programming problems.

Programs B17a(c).1 and B17a(c).2 in the Program Appendix meet the requirements of the present objective for addition and multiplication.

Multiple-Precision Arithmetic

At least two distinct approaches to doing multiple-precision arithmetic are common:

(a) using arrays and modular arithmetic
(b) using strings

The first method is best applied to a problem such as the calculation of N!, where each factor of the product is relatively small, although the final result may be quite large. Suppose it is desired to find 13!, where 12! has already been calculated. Then 13! = 12! * 13

$$= \underset{S(7)}{\underline{2}}\ \underset{S(6)}{\underline{432}}\ \underset{S(5)}{\underline{902}}\ \underset{S(4)}{\underline{008}}\ \underset{S(3)}{\underline{176}}\ \underset{S(2)}{\underline{640}}\ \underset{S(1)}{\underline{000}} * 13$$

The array element S(1) is used to store the least significant three digits 000 (=12! (mod 1000)).

S(2) is used to store the next three least significant digits 640, and so on. The multiplier 13 is multiplied into each of the S(I) by ordinary computer arithmetic, and the results are added in such a way as to give new values for the S(I) which are now the digits of 13! When this approach is applied to adding (subtracting), each number is divided into groups of three digits as explained above, corresponding groups are added (subtracted), and the resultant sum adjusted appropriately for digits carried (borrowed) from one group to another.

This approach is used to calculate N! in the book Problems for Computer Solution, Teacher Edition, by Stephen J. Rogowski.

The approval of strings, perhaps more flexible although probably slower to run, is used in the program B17a(c) in the Program Appendix. Each number is input as a string. This guarantees that the number is not rounded off when entered into memory.

Consider first the addition problem. If the numbers are not integers each string can be scanned to locate the decimal points. Once they are located, each number can be adjusted, by adding zeros, to the same number of digits after the decimal point. The problem, at this point, is simply one of adding integers, since the decimal point can be inserted into the final answer at the appropriate point. To add integers, first the units digits are added, then the ten (adding one for the carry, if any), then the hundreds, ets. The result of each addition is stored as a character in a new string (by using VAL and STRS functions). The new string, when completed, contains numerical characters which form the sum of the original two number-strings. Since it is a string rather than a number, however, it can be printed in its entirety without round off.

Subtraction works essentially the same way, except that rather than having a "carry" to add to the next sum it may be necessary to subtract a "borrow" from that difference.

Multiplication is also approached on a digit-by-digit basis, where it is necessary to keep track of carries. As an example, consider 789 x 456.

1's x 1's : 9 x 6 = 54 . Keep $\underline{4}$, carry 5.
1's x 10's + 10's x 1's : 48 + 45 + 5 = 98. Keep $\underline{8}$, carry 9.
1's x 100's +10's x 10's + 100's x 1's : 42 + 40 + 36 + 9 = 127. Keep $\underline{7}$, carry 12.
10's x 100's + 100's x 10's : 35 + 32 + 12 = 79. Keep $\underline{9}$, carry 7.
100's x 100's : 28 + 7 = 35. Keep $\underline{5}$, carry 3.
No further products : 0 + 3 = 3. Keep $\underline{3}$, carry 0.

Assembling the kept digits from right to left, the result 359784 is the desired product. Clearly this method will apply to numbers of arbitrary length. Since each of he kepot digits is stored as one character of a string in the required BASIC multiplication program, there is no round off involved in printing the final answer Program B17a(c).1 gives a solution to the addition problem, and in fact allows the addition of several integers. It can easily be extended, as outlined above, to addition of decimal fractions.

Program B17a(c).2 gives a solution to the multiplication problem of either integers or decimal fractions. It allows for chaining of multiplications, by being able to use the product of one multiplication as a factor of the next one.

No program has been included for the subtraction problem, but anyone who has solved the addition problem should have no difficulty adapting his program to subtraction.

Interested mathematically capable students may wish to attempt the problem of multiple-precision division. No program has been included for this problem.

COMPUTER MATHEMATICS

Objective: B17a(d) - The student will print a calendar for any given year.

Reference: Martin Gardner, "Mathematical Games," Scientific American, May 1967.
Steven J. Rogowski, Problems for Computer Solutions, Teacher Edition, Creative Computing Press.
BASIC Computer Games, David Ahl, Editor, Workman Publishing, 1978.

TEACHING SUGGESTIONS	STUDENT ACTIVITIES
This is a problem that will probably require some research work on the part of the student. A good place to start is with Martin Gardner's Scientific American article. BASIC programs for perpetual calendars are contained inthe second and third references listed above. Obviously if this objective is used as a major project, the student should do more than copy a ready-made program. Strictly speaking, the program in the Rogowski reference does not print a calendar but simply identifies the day of the week, given the month, date, and year. The Ahl program prints a true calendar for a given year, but must be given the day of the week for January 1. Possibly the student could combine these two ideas, so that, given only the year, the program could determine the starting day and proceed to print out a correct calendar for that year.	

COMPUTER MATHEMATICS

Objective: B17a(e) - The student will compute n! and compare the exact value to Stirling's approximation.

Reference: CRC Standard Mathematical Tables

TEACHING SUGGESTIONS

The factorial function n! has been expanded as an infinite series:

$$n! = \sqrt{2\pi n}\; n^n e^{-n} \left(1 + \frac{1}{12n} + \ldots\right)$$

Stirling's approximation is obtained by keeping ly the term 1 in the series, thus

n! is approximately equal to
$F(n) = \sqrt{2\pi n}\; n^n e^{-n}$

Also,

$F(n) \times \frac{12n+1}{12n}$, although not usually used, is an even more accurate approximation.

(e is the base of natural logorithms, e = 2.71828...
The function e^{-n} is found on the computer as EXP(-N).)

Comment on Problem 4:

The gamma function is a mathematical extension of the concept of factorial to all real numbers (except the negative integers, where it is undefined).

The theory shows that

$$(1,5)! = \frac{3}{4}\sqrt{\pi} \text{ precisely.}$$

Program B17a(e) in the Program Appendix uses the ordinary approximation for N from 1 to 15.

STUDENT ACTIVITIES

1. Stirling's approximation improves as N gets larger.

   Find the smallest value of N for which Stirling's approximation is accurate to better than

   a) 10%
   b) 1%

2. Calculate F(N)/N! for N going from 1 to 15.

3. Calculate $\frac{12N+1}{12N}$ F(N)/N!

   for N = 1 and N = 10.

4. (Advanced) The factorial concept can be extended to integers. Use Stirling's approximation, including the $\frac{12N+1}{12N}$ factor, to estimate the value of (1.5)! How close is this value to $\frac{3}{4}\sqrt{\pi}$?

Answers:

1. a) 1 b) 9

2. Sample values

   1 - .922137
   10 - .991705

3. 1 - .998982
   10 - .999969

4. 1.32835
   1.32934

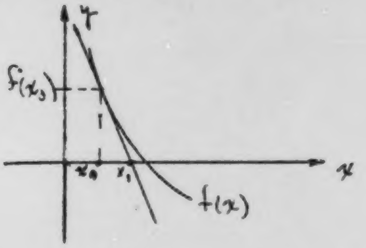
COMPUTER MATHEMATICS

Objective: B17a(f) - The student will find the zeros of a polynomial by using Newton's Method.

Reference: Discussion of Objectives B7 and B7a in Topic 14

TEACHING SUGGESTIONS

As mentioned in the discussion of Objectives B7 and B7a, Newton's method can be used to find iterative formulas for use in a converging loop. In fact, finding solutions to the equation f(x)=0 by Newton's method is nothing more, once a proper iterative formula is known, that setting up an iterative loop.



From the figure, x, should be closer to the zero of f(x) than is the initial guess xo. The slope of the tangent line is

$$\frac{f(x_0)}{x_1-x_0}$$ and the slope of f(x)

is $f^1(x_o)$, where $f^1(x)$ is the derivative function. Since the slopes are the same,

$$f^1(x_o) = -\frac{f(x_o)}{x_1-xo},$$
or
$$x_1 = x_0 - \frac{f(x_o)}{f^1 x_o)}$$, which is Newton's approximation.

Programs B17a(f) in the Program Appendix illustrate x^3-4x+2=0 and x = cos(x) by applying Newton's Method.

STUDENT ACTIVITIES

Use Simpson's Rule to solve -

a) $x^3 - 4x + 2 = 0$

b) x = cos(x), where x must be given in radion measure

Answer:

a) -2.21432, 0.539189, and 1.67513

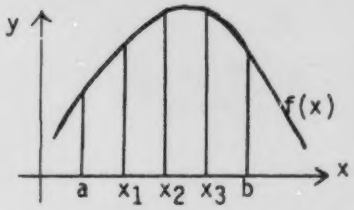
b) 0.739085

COMPUTER MATHEMATICS

Objective: B17a(g) - The student will write a BASIC program to find the area under a curve by using Simpson's Rule.

Reference: CRC Standard Mathematical Tables

TEACHING SUGGESTIONS



The area under a curve f(x) over the internal (a,b) can be approximated by dividing the interval into a number not equally spaced subintervals, drawing straight lines from f(a) to $f(x_1)$, $f(x_1)$ to $f(x_2)$,..., $f(x_{n-1})$ to f(b), and adding up the areas under each of the trapezoids formed in this manner. Increasing the number of subintervals increases the accuracy of the approximation. Such a method is known as a Trapezoidal Rule.

An even more accurate procedure is to choose an even number of subintervals and, taking them two at a time, to fit a parabola to the two endpoints and the centerpoint. The formula for the result is remarkably simple, and is known as Simpson's Rule. The rule states that the area is approximately

$$\frac{b-a}{3n}\ [f(a)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\ldots+4f(x_{n-1})+f(b)]$$

Warning: For this to work the number of intervals n must be even.

STUDENT ACTIVITIES

1. Use the trapezoidal rule to calculate by hand the area under the curve $y = x^2$ over the interval 0 to 1. Use 4 intervals.
2. Use Simpson's rule to calculate by hand the area under the curve $y = x^2$ over the interval 0 to 1. Use 4 intervals.
3. Repeat Problem Two for the curve $y=x^3$.
4. Repeat Problems 2 and 3 using computer program.

Answers:

1. $\frac{1}{4}(\frac{1}{32} + \frac{5}{32} + \frac{13}{32} + \frac{25}{32}) = \frac{11}{32}$
2. $\frac{1}{12}\ (0 + \frac{1}{4} + \frac{1}{2} + \frac{9}{4} + 1) = \frac{1}{3}$
3. $\frac{1}{12}\ (0 + \frac{1}{16} + \frac{1}{4} + \frac{27}{10} + 1) = \frac{1}{4}$
4. 1/3, 1/4

TEACHING SUGGESTIONS

In Program B17a(g).1 of the Program Appendix, Simpson's Rule is used to find the area under the function $\frac{1}{X^2 + X}$ over the interval 1 to 2. Note that the answer has already been approximated to 4 significant figures for n = 4.

Programs B17a(g).2 and B17a(g).3 are examples of how Simpson's Rule can be written as a subroutine--the first in BASIC and the second in FORTRAN. The function to be integrated needs to be defined and inserted into the subroutine.

STUDENT ACTIVITIES

COMPUTER MATHEMATICS

Objective: B17a(i) - The student will generate and list prime numbers by using the sieve method.

Reference: Scientific American, December, 1958, pp. 105-112

TEACHING SUGGESTIONS

A sieve is a fine mesh used to sift small items away from larger ones. The method used for sifting primes is sometimes called the sieve of Eratosthenes.

The student activity listed at right shows the method, along with hints for writing a program. The student should learn a lot by actually generating primes up to 400 by this method. He will also have a list to check his computer program results against. Improperly-treated roundoff errors can introduce spurious answers into a list of supposed primes. (See Objective B3, under the Additional Objectives.)

Programs B17a(i) are efficient programs, based on the sieve method, for generating primes. The first lists primes from 2 up to N and the second from N1 to N2.

Once a person gains a resonable amount of programming skill it becomes important to stress that programs performing a long calculation should do so efficiently.

The prime number program A17(c).1 is easily understood by students but takes a long time to generate primes above 400. The sieve program is more complex and more difficult for all students to understand but it is fast.

STUDENT ACTIVITIES

Write in rows of 20 each, the integers from 1 to 400.

a) Cross out 1. 1 is not considered to be prime.
b) Cross out all multiples of 2 except 2 itself.
c) Cross out all multiples of 3 except 3 itself.
d) The smallest number not yet crossed out is 5. Cross out all multiples of 5 except 5 itself. (Note that the smallest not yet crossed out is 7.
e) The smallest number not yet crossed out is 7. Cross out all multiples of 7 except 7 itself. (Note that the smallest not yet crossed out is $7^2 = 49$.)
f) Continue this process, until all possible multiples have been crossed out. (Note that multiples of 19 were the last ones to be crossed out, and that 19 is the largest prime less than the square root of 400.)

The numbers which are not yet crossed out are the primes between 1 and 400.

COMPUTER MATHEMATICS

Objective: B17a(j) - The student will use a sorting procedure more efficient than the bubble sort to sort (alphabetize) numerical (alphabetical) data.

Reference: Thomas Dwyer and Margot Critchfield, BASIC and the Personal Computer, Addison-Wesley Publishing Company (1979), Chapter 5.

TEACHING SUGGESTIONS

The sorting procedure on page 282 of the textbook is called a bubble sort (see Objective A17(m) discussion, Topic 9). The small numbers bubble to the top of the list. (Actually, as usually written, the large numbers sink to the bottom.)

When applied to a large set of random numbers a bubble sort proves not to the very efficient. Time spent on a computer costs a company money. It is certainly worthwhile to try to reduce calculation time by using efficient algorithms for solution of computer problems.

Two sorting algorithms which have been developed to improve sorting efficiency are called Shellsort and Quicksilver. They are both carefully discussed, along with other alternatives in the Dwyer-Critchfield textbook listed above as reference. (This textbook is excellent, and well worth adding to both personal and professional libraries.)

Programs B17a(j).1 and B17a(j).2 provide examples of Shellsort and Quicksort applied to the problem of alphabetization. See Supplement B17a(j) - Sorting Algorithms.

STUDENT ACTIVITIES

1. Review the bubblesort algorithm, page 282 of textbook. Hand-run a bubblesort of the list 6,1,17,3,7,4,8,10,5,2.
   Keep track of the number or comparisons and exchanges needed to complete the sorting procedure.
2. Hand-run a Shellsort in the list of of problem 1. Keep track of the number of comparisons and exchange needed to complete the sorting procedure.
3. Same as problem 2, using Quicksort.
4. Determine the length of time needed for a program to sort a list of 500 random integers (from 0 to 999)

   a) by bubblesort
   b) by Shellsort
   c) by Quicksort

   How did the times compare?

   Answer: 1-3, On Supplement B17a(j).
5. Typical answers for a microcomputer with a BASIC interpreter would be

   a) 18-1/2 minutes
   b) 165 seconds
   c) 115 seconds

Supplement B17a(j)

Sorting Algorithms

In this supplement comparisons will be made among three sorting algorithms: bubblesort, Shellsort, and Quicksort. The problem is to sort N numbers from smallest to largest.

Bubblesort always compares neighboring numbers in the list, interchanging them if they are out of order. For a list of N numbers the first pass requires N-1 comparisons. At the end of the first pass the largest number is in proper sorted position. A second pass of N-2 comparisons, a third of N-3, etc., are performed. If, on a given pass, no interchanges are made, the list has been sorted and the algorithm ends.

Shellsort is similar to bubblesort except that sorting begins by comparing numbers separated by half the length of the list. Thus N/2 comparisons are made during the first pass. As with bubblesort, numbers out of order are interchanged. For the second pass, numbers separated by one-fourth the length of the list are compared by one-eighth the length of the list are compared, and so on, until finally a simple bubblesort is being performed. However, since by this time the list has been presorted, the bubblesort may not require many interchanges.

Quicksort is somewhat more complicated to explain and to program. The first number in the list is designated as a "supernumber," and the last as a pointer. If out of order, they are interchanged. Suppose that the supernumber is 6 and the pointer 2. Then the two will be interchanged so that 2 is first and 6 is last. However 6 remains the supernumber and 2 the pointer. Next, the number beside the new position of 2 and closest to the 6 becomes the new pointer. The supernumber and the new pointer are compared, and interchanged, if necessary. The process continues until the pointer and the supernumber are adjacent, at which time the supernumber is in final sorted position, somewhere inside the list. The first pass has ended, and the list has been divided into two smaller sublists. The process continues, first with the left-hand sublist. After each pass has ended, the leftmost number of the sublist is the new supernumber and the rightmost, initially, is the pointer. The process continues until all sublists, and thus the original list, are sorted.

Since it is easier to carry out the sorting process than to read about it, each algorithm will be illustrated by using it to sort the sequence 6,1,17,3,7,4,8,10,5,2. Record will be kept of the numbers of comparisons and interchanges performed, as an indication of the efficiency of each algorithm.

The following results are obtained for this sequence:

Type of sort	No. of comparisons	No. of Interchanges	C + 3I
Bubble	45	22	111
Shell	37	14	79
Quick	20	10	50

An interchange requires three separate statements, for example, T = A: A = B : B = T. Thus C + 3I, the number of comparisons plus three times the number of interchanges, is a sort of figure of merit for efficiency. Clearly for the present sequence, Quicksort should be fastest and bubblesort slowest. For longer sequences the differences can be even more dramatic. As a word of caution, though, for sequences which are almost sorted to begin with, bubblesort may turn out to be the fastest of the three.

Bubblesort

```
                                                             C   I
6     1     17    3     7     4     8     10    5     2
1     6                                                      1   1
            3     17                                         2   1
                  3     17                                   1   1
                        4     17                             1   1
                              8     17                       1   1
                                    10    17                 1   1
                                          5     17           1   1
                                                2     17     1   1
1     6     3     7     4     8     10    5     2     17     9   8   End of 1st pass; 17 is sorted

      3     6                                                2   1
            4     7                                          2   1
                                    5     10                 3   1
                                          2     10           1   1
1     3     6     4     7     8     5     2     10    17     8   4   End  of 2nd pass; 10 is sorted

            4     6                                          3   1
                              5     8                        3   1
                                    2     8                  1   1
1     3     4     6     7     5     2     8     10    17     7   3   End of 3rd pass; 8 is sorted

                  5     7                                    5   1
                        2     7                              1   1
1     3     4     6     5     2     7     8     10    17     6   2   End of 4th pass; 7 is sorted

                  5     6                                    4   1
                        2     6                              1   1
1     3     4     5     2     6     7     8     10    17     5   2   End of 5th pass; 6 is sorted

                  2     5                                    4   1
1     3     4     2     5     6     7     8     10    17     4   1   End of 6th pass; 5 is sorted

            2     4                                          3   1
1     3     2     4     5     6     7     8     10    17     3   1   End of 7th pass; 4 is sorted

      2     3                                                2   1
1     2     3     4     5     6     7     _     10    17     2   1   End of 8th pass; 3 is sorted

1     2     3     4     5     6     7     8     10    17     1   0   End of 9th pass; no interchanges.
                                                                     Sorting has ended.
```

Sorting required 45 comparisons, 22 interchanges.

Shellsort

```
                                                               C    I
6     1     17    3     7     4     8     10    5     2
4                             6                                1    1   Compare 5 numbers apart
            10                                  17             2    1
                        2                             7        2    1
4     1     10    3     2     6     8     17    5     7        5    3   End of 1st pass.

            2           10                                     3    1   Compare INT(5/2)=2 numbers apart
                        8           10                         2    1
                                    5           10             2    1
                                          7           17       1    1
4     1     2     3     8     6     5     7     10    17       8    4   End of 2nd pass.

1     4                                                        1    1   Compare INT(2/2)=1 numbers apart
      2     4                                                  1    1   This is bubble sort.
            3     4                                            1    1
                        6     8                                2    1
                              5     8                          1    1
                                    7     8     10    17       3    1
1     2     3     4     6     5     7     8     10    17       9    6   End of 3rd pass; 17 is sorted

                        5     6                                8    1
1     2     3     4     5     6     7     8     10    17       8    1   End of 4th pass; 10 is sorted

1     2     3     4     5     6     7     8     10    17       7    0   End of 5th pass; no interchanges.
                                                                        Sorting has ended.
```

Sorting required 37 comparisons, 14 interchanges.

Quicksort

										C	I	
$\underline{6}$	1	17	3	7	4	8	10	5	2			6 is the supernumber (underlined)
2									$\underline{6}$	1	1	2 is the pointer, then 1
		$\underline{6}$							17	2	1	17 is the pointer
		5						$\underline{6}$		1	1	5 is the pointer
				$\underline{6}$				7		2	1	7 is the pointer
				4	$\underline{6}$					$\underline{3}$	$\underline{1}$	
2	1	5	3	4	$\underline{6}$	8	10	7	17	9	5	End of 1st pass; 6 is sorted.
$\underline{2}$	1	5	3	4								Begin sorting next leftmost/sublist
1	$\underline{2}$									$\underline{4}$	$\underline{1}$	2 is the supernumber, 4 the pointer
$\underline{1}$	$\underline{2}$	5	3	4						4	1	End of 2nd pass; 2 is sorted
$\underline{1}$	$\underline{2}$	5	3	4	$\underline{6}$	8	10	7	17			1 is sorted
		$\underline{5}$	3	4								Begin sorting next leftmost/sublist
		4		$\underline{5}$						$\underline{2}$	$\underline{1}$	5 is the supernumber; 4 is the pointer
$\underline{1}$	$\underline{2}$	4	3	$\underline{5}$	$\underline{6}$	8	10	7	17	2	1	End of 3rd pass; 5 is sorted
		$\underline{4}$	3									4 is the supernumber; 3 is the pointer
		3	$\underline{4}$							$\underline{1}$	$\underline{1}$	
$\underline{1}$	$\underline{2}$	3	4	$\underline{5}$	$\underline{6}$					1	1	End of 4th pass; 4 is sorted
$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	8	10	7	17			3 is sorted
						$\underline{8}$	10	7	17			Begin sorting remaining sublist.
						7		$\underline{8}$		2	1	8 is the supernumber, 17 the pointer
							$\underline{8}$	10		$\underline{1}$	$\underline{1}$	10 is the pointer
						7	$\underline{8}$	10	17	3	2	End of 5th pass; 8 is sorted
$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	$\underline{8}$	10	17			7 is sorted
								$\underline{10}$	17	1	0	Begin sorting remaining sublist
												10 is the supernumber, 17 the pointer
$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	$\underline{8}$	$\underline{10}$	$\underline{17}$			10 is sorted. 17 is sorted.
												Sorting has ended.

Sorting required 20 comparisons, 10 interchanges.

Shellsort

										C	I	
6	1	17	3	7	4	8	10	5	2			
4					6					1	1	Compare 5 numbers apart
		10					17			2	1	
				2					7	2	1	
4	1	10	3	2	6	8	17	5	7	5	3	End of 1st pass.
		2		10						3	1	Compare INT(5/2)=2 numbers apart.
				8		10				2	1	
						5		10		2	1	
							7		17	1	1	
4	1	2	3	8	6	5	7	10	17	8	4	End of 2nd pass.
1	4									1	1	Compare INT(2/2)=1 numbers apart.
	2	4								1	1	This is bubble sort.
		3	4							1	1	
				6	8					2	1	
					5	8				1	1	
						7	8			3	1	
1	2	3	4	6	5	7	8	10	17	9	6	End of 3rd pass; 17 is sorted.
				5	6					8	1	
1	2	3	4	5	6	7	8	10	17	8	1	End of 4th pass; 10 is sorted.
1	2	3	4	5	6	7	8	10	17	7	0	End of 5th pass; no interchanges. Sorting has ended.

Sorting required 37 comparisons, 14 interchanges.

```
10 REM SHELLSORT TO SORT A LIST OF N NUMBERS
20 READ N
30 FOR I = 1 TO N
40 READ A(I)
50 NEXT I
60 DATA 10,6,1,17,3,7,4,8,10,5,2
70 REM IN ASCENDING ORDER
80 LET S = N
90 LET S = INT(S/2)
100 IF S<1 THEN 230
110 FOR K = 1 TO S
120 FOR I = K TO N-S STEP S
130 LET J = I
140 LET T = A(I+S)
150 IF T>=A(J) THEN 190
160 LET A(J+S)=A(J)
170 LET J = J-S
180 IF J>=1 THEN 150
190 LET A(J+S) = T
200 NEXT I
210 NEXT K
220 GOTO 90
230 FOR I = 1 TO N
240 PRINT A(I);
250 NEXT I
260 END
```

```
 1  2  3  4  5  6  7  8  10  17
```

Quicksort

										$\underline{C}$	$\underline{I}$	
6	1	17	3	7	4	8	10	5	2	—	—	6 is the supernumber (underlined)
$\overline{2}$									6	1	1	2 is the pointer, then 1
		6							$\overline{17}$	2	1	17 is the pointer
		$\overline{5}$						6		1	1	5 is the pointer
				6				$\overline{7}$		2	1	7 is the pointer
				$\overline{4}$	6					3	1	
2	1	5	3	4	$\overline{\underline{6}}$	8	10	7	17	$\overline{9}$	$\overline{5}$	End of 1st pass; 6 is sorted.
2	1	5	3	4								Begin sorting leftmost sublist.
$\overline{1}$	2									4	1	2 is the supernumber, 4 the pointer.
1	$\overline{2}$	5	3	4						$\overline{4}$	$\overline{1}$	End of 2nd pass; 2 is sorted.
$\underline{1}$	$\underline{\overline{2}}$	5	3	4	$\underline{6}$	8	10	7	17			1 is sorted.
		5	3	4								Begin sorting next leftmost sublist
		$\overline{4}$		5						2	1	5 is the supernumber; 4 is the pointer
$\underline{1}$	$\underline{2}$	4	3	$\underline{\overline{5}}$	$\underline{6}$	8	10	7	17	$\overline{2}$	$\overline{1}$	End of 3rd pass; 5 is sorted.
		4	3									4 is the supernumber; 3 is the pointer
		$\overline{3}$	4							1	1	
1	2	3	$\overline{4}$	5	6					$\overline{1}$	$\overline{1}$	End of 4th pass; 4 is sorted.
$\underline{\overline{1}}$	$\underline{\overline{2}}$	$\underline{3}$	$\underline{\overline{4}}$	$\underline{\overline{5}}$	$\underline{\overline{6}}$	8	10	7	17			3 is sorted.
						8	10	7	17			Begin sorting remaining sublist.
						$\overline{7}$		8		2	1	8 is the supernumber, 17 the pointer
							8	$\overline{10}$		1	1	10 is the pointer.
						7	$\overline{8}$	10	17	$\overline{3}$	$\overline{2}$	End of 5th pass; 8 is sorted.
$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	$\underline{\overline{8}}$	10	17			7 is sorted.
								$\underline{10}$	17	1	0	Begin sorting remaining sublist. 10 is the supernumber, 17 the pointer.
$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	$\underline{8}$	$\underline{10}$	$\underline{17}$			10 is sorted. 17 is sorted. Sorting has ended.

Sorting required 20 comparisons, 10 interchanges.

```
 READ N
10 REM QUICKSORT
20 FOR I = 1 TO N
30 READ A(I)
40 NEXT I
50 DATA 10,6,1,17,3,7,4,8,10,5,2
60 L = 1
70 B(L) = N+1
80 M = 1
81 REM SET END OF ARRAY SEGMENT
85 J = B(L)
90 REM SET START OF ARRAY SEGMENT
100 I = M-1
110 REM IF ONLY 1 TO 2 ELEMENTS REMAIN HANDLE THEN SPECIALLY
120 IF J - M < 3 THEN 330
130 M1 = INT((I+J)/2)
131 REM FIND A BIG ELEMENT AMONG THE SMALL ONES
140 I = I+1
150 IF I = J THEN 250
155 IF A(I) <= A(M1) THEN 140
156 REM FIND A SMALL ELEMENT AMONG THE BIG ONES
160 J = J-1
170 IF I = J THEN 250
180 IF A(J)>=A(M1) THEN 160
181 REM EXCHANGE OUT OF PLACE ELEMENTS
190 T = A(I)
200 A(I) = A(J)
210 A(J) = T
230 GOTO 140
240 REM ARRAY SEGMENT NOW DIVIDED
250 IF I>= M1 THEN I = I-1
260 IF J = M1 THEN 290
265 T = A(I)
270 A(I) = A(M1)
280 A(M1) = T
285 REM SAVE STARTING POINT FOR ARRAY  SEGMENT OF BIG ELEMENTS
290 L = L+1
300 B(L) = I
310 REM SAVE STARTING POINT FOR SMALL ARRAY
320 GOTO 85
330 IF J - M < 2 THEN 380
340 IF A(M) < A(M+1) THEN 380
345 T = A(M)
350 A(M) = A(M+1)
360 A(M+1) = T
370 REM SET POINTS FOR ARRAY SEGMENT OF BIG ELEMENTS
380 M = B(L) + 1
390 L = L-1
400 IF L> 0 THEN 85
410 FOR I = 1 TO N
420 PRINT A(I);
430 NEXT I
```

```
 1  2  3  4  5  6  7  8  10  17
```

```
5 READ N
10 FOR I = 1 TO 10
20 RANDOMIZE I
30 READ A(I)
40 NEXT I
50 REM BINARY SEARCH TO SEARCH LIST A OF LENGTH N FOR V.
60 REM M IS SET EQUAL TO THE INDEX I FOR WHICH A(I) = V.
70 REM M = 0  MEANS V WAS NOT FOUND.
80 LET L = 1
90 R = N
100 INPUT "WHAT IS THE NUMBER YOU ARE LOOKING FOR";V
110 LET M = INT((L+R)/2)
120 IF V = A(M) THEN 190
130 IF V < A(M) THEN 160
140 LET L = M+1
150 GOTO 170
160 LET R = M-1
170 IF L<=R THEN 110
180 LET M = 0
190 LPRINT V "IS THE NUMBER SEARCHED FOR"
200 LPRINT "A("M") IS ITS LOCATION"
210 LPRINT "THE ARRAY IS";
220 FOR I = 1 TO N
230 LPRINT A(I);
240 NEXT I
250 DATA 10,1,2,3,4,5,6,7,8,10,17
```

```
 7 IS THE NUMBER SEARCHED FOR
A( 7 ) IS ITS LOCATION
THE ARRAY IS 1  2  3  4  5  6  7  8  10  17
 10 IS THE NUMBER SEARCHED FOR
A( 9 ) IS ITS LOCATION
THE ARRAY IS 1  2  3  4  5  6  7  8  10  17


 17 IS THE NUMBER SEARCHED FOR
A( 10 ) IS ITS LOCATION
THE ARRAY IS 1  2  3  4  5  6  7  8  10  17
```

PROGRAM APPENDIX

1. B3 - Pitfalls of Computer Arithmetic
2. B4 - Interbase Number Conversion
3. B6 - Matrix Multiplication Without MAT Multiplication
4. B6a - Matrix Inversion
5. B6* - Value of a Rank-N Determinant
6. B9a - Compucolor Files
7. B13a - PEEK and POKE
8. B14 - String Functions
9. B17a(a) - Addition and Subtraction of Fractions
10. B17a(b) - Cube Roots of a Real Number
11. B17a(c).1 - Multiple-precision Addition
12. B17a(c).2 - Multiple-precision Multiplication
13. B17a(e) - Stirling's Approximation to N!
14. B17a(f).1 - Newton's Method for Polynomial
15. B17a(f).2 - Newton's Method for Trigonometric Function
16. B17a(g).1 - Integration by Simpson's Rule
17. B17a(g).2 - BASIC Subroutine, Simpson's Rule
18. B17a(g).3 - FORTRAN Subroutine, Simpson's Rule
19. B17a(i).1 - Prime Numbers, Sieve Method
20. B17a(i).2 - Prime Numbers, Sieve Method - Extended Range
21. B17a(j).1 - Shellsort Sorting Program
22. B17a(j).2 - Quicksort Sorting Program

1. B3

```
10 REM   THIS PROGRAM SHOWS SOME PITFALLS OF DOING COMPUTER
20 REM   ARITHMETIC.
25 PRINT "IS 5 THE SQUARE ROOT OF 25? (USE TEST IN LINE 30.)"
30 IF SQR (25)= 5THEN PRINT "YES, IT IS!":GOTO 50
40 PRINT "SORRY! NOT ON THIS COMPUTER.  DO YOU WANT TO TRY AGAIN
?"
45 PRINT :PRINT "IS 5 THE SQUARE ROOT OF 25? (USE TEST IN LINE 5
0.)"
50 IF ABS (SQR (25)- 5)) 1E- 5THEN PRINT "NO, IT STILL ISN'T":GO
TO  70
60 PRINT "GOOD!  THIS TIME YOU USED A TOLERANCE. YES, 5 IS THE S
QUARE ROOT OF 25."
65 PRINT
70 PRINT "NOW LET'S TRY SOME MULTIPLICATION."
80 PRINT "WHAT IS 1E-30 X 1E-30 X 1E30 X 1E30 ?  LET'S ASK THE C
OMPUTER."
85 PRINT
90 PRINT "THE ANSWER IS";1E- 30* 1E- 30* 1E30* 1E30;SPC( 5);"(NO
 PARENTHESES)"
95 PRINT :PRINT "THE PROBLEM IS, THAT 1E-60 'UNDERFLOWS' TO 0: "
100 PRINT "THE COMPUTER CAN'T HANDLE NUMBERS THAT SMALL."
110 PRINT "SO LET'S TRY GROUPING WITH PARENTHESES."
115 PRINT
120 PRINT "THE ANSWER REALLY IS";1E- 30* (1E- 30* 1E30)* 1E30
130 PRINT "THIS TIME THE FACTORS WERE ASSOCIATED (COMPUTER "
135 PRINT "MULTIPLICATION MAY NOT BE ASSOCIATIVE!) IN ORDER TO"
140 PRINT "AVOID UNDERFLOW."
145 PRINT
150 PRINT "WHAT IF WE PRINT  1E-30*1E-30*(1E30*1E30) ?"
155 PRINT "WE WOULD GET AN OVERFLOW ERROR, STOPPING ALL OUTPUT,"
160 PRINT "SO I'M NOT GOING TO DO UNTIL THE END OF THE PROGRAM."
175 PRINT
180 PRINT "HERE'S AN EASY ONE!  WHAT IS 3↑2 ?"
190 IF 3↑ 2= 9THEN PRINT "3↑2 =9":GOTO 210
200 PRINT "3↑2 ISN'T 9 BECAUSE 3↑2 - 9 IS ";3↑ 2- 9
210 PRINT
220 PRINT "ROUNDOFF ERROR THIS TIME ON THE EXPONENT OPERATION."
225 PRINT "REASONABLE, IF YOU THINK ABOUT HOW YOU WOULD CALCULAT
E"
230 PRINT " 4.7↑2.6 .  CAN YOU DO IT EXACTLY?"
235 PRINT "YOU'RE BETTER OFF WITH 3*3, BECAUSE EVEN THE COMPUTER
"
237 PRINT "KNOWS THAT 3*3 IS EXACTLY";:IF 3* 3= 9THEN PRINT 9
238 PRINT
240 PRINT "AS A FINAL EXAMPLE, LET'S EVALUATE THE FUNCTION
250 PRINT "X↑3 - 6X↑2 + 4X - 0.1157  AT X=5.241"
260 PRINT "WE'LL WRITE X↑3 AS X*X*X JUST IN CASE OF ROUNDOFF ERR
OR"
265 X= 5.241:PRINT
270 PRINT "THE ANSWER IS";X* X* X- 6* X* X+ 4* X- 0.1157
280 PRINT "UNFORTUNATELY, AS YOU CAN VERIFY BY DIRECT CALCULATIO
N,
290 PRINT "THE CORRECT ANSWER IS 2.6521 E-5."
295 PRINT
300 PRINT "PARENTHESES CAN HELP, SINCE ((X-6)X +4)X -0.1157 IS";
310 PRINT ((X- 6)* X+ 4)* X- 0.1157;"."
320 PRINT "BUT DOUBLE PRECISION CALCULATION WOULD HELP A LOT "
330 PRINT "MORE.  UNFORTUNATELY MANY COMPUTERS (INCLUDING THIS"
340 PRINT "ONE), DON'T USE DOUBLE OR EXTENDED PRECISION."
345 PRINT
350 PRINT "OH, YES.  FINALLY, WE NOTE THAT 1E-30*1E-30*(1E30*1E3
0) IS EQUAL TO"
360 PRINT 1E- 30* 1E- 30* (1E30* 1E30)
```

1. B3 - Pitfalls of computer arithmetic. (page 2)

```
RUN

IS 5 THE SQUARE ROOT OF 25? (USE TEST IN LINE 30.)
SORRY! NOT ON THIS COMPUTER.  DO YOU WANT TO TRY AGAIN?

IS 5 THE SQUARE ROOT OF 25? (USE TEST IN LINE 50.)
GOOD!  THIS TIME YOU USED A TOLERANCE. YES. 5 IS THE SQUARE ROOT
 OF 25.

NOW LET'S TRY SOME MULTIPLICATION.
WHAT IS 1E-30 X 1E-30 X 1E30 X 1E30 ?  LET'S ASK THE COMPUTER.

THE ANSWER IS 0      (NO PARENTHESES)

THE PROBLEM IS. THAT 1E-60 'UNDERFLOWS' TO 0:
THE COMPUTER CAN'T HANDLE NUMBERS THAT SMALL.
SO LET'S TRY GROUPING WITH PARENTHESES.

THE ANSWER REALLY IS 1
THIS TIME THE FACTORS WERE ASSOCIATED (COMPUTER
MULTIPLICATION MAY NOT BE ASSOCIATIVE!) IN ORDER TO
AVOID UNDERFLOW.

WHAT IF WE PRINT  1E-30*1E-30*(1E30*1E30) ?
WE WOULD GET AN OVERFLOW ERROR, STOPPING ALL OUTPUT,
SO I'M NOT GOING TO DO UNTIL THE END OF THE PROGRAM.

HERE'S AN EASY ONE!  WHAT IS 3↑2 ?
3↑2 ISN'T 9 BECAUSE 3↑2 - 9 IS  2.86102E-06

ROUNDOFF ERROR THIS TIME ON THE EXPONENT OPERATION.
REASONABLE. IF YOU THINK ABOUT HOW YOU WOULD CALCULATE
 4.7↑2.5 .  CAN YOU DO IT EXACTLY?
YOU'RE BETTER OFF WITH 3*3. BECAUSE EVEN THE COMPUTER
KNOWS THAT 3*3 IS EXACTLY 9

AS A FINAL EXAMPLE. LET'S EVALUATE THE FUNCTION
X↑3 - 6X↑2 + 4X - 0.1157  AT X=5.241
WE'LL WRITE X↑3 AS X*X*X JUST IN CASE OF ROUNDOFF ERROR

THE ANSWER IS 1.69426E-05
UNFORTUNATELY. AS YOU CAN VERIFY BY DIRECT CALCULATION.
THE CORRECT ANSWER IS 2.6521 E-5.

PARENTHESES CAN HELP. SINCE ((X-6)X +4)X -0.1157 IS 1.9744E-05,
BUT DOUBLE PRECISION CALCULATION WOULD HELP A LOT
MORE.  UNFORTUNATELY MANY COMPUTERS (INCLUDING THIS
ONE). DON'T USE DOUBLE OR EXTENDED PRECISION.

OH. YES.  FINALLY. WE NOTE THAT 1E-30*1E-30*(1E30*1E30) IS EQUAL
 TO

OV ERROR IN 360

READY
```

2. B4 - Interbase number conversion.
This program converts between any two number bases from 2 to 36, and should work for any system that recognizes the ASC and CHR$ statements. For base 36 the symbols used are the numbers 0-9 and letters A-Z. This program can be used to code English text into numbers. For example, HOW$_{36}$ ARE$_{36}$ YOU$_{36}$ has base-10 equivalent 22928 13946 44958.

```
0 REM  - COMP. MATH. B, OBJECTIVE 4.  NO PROGRAM IS REQUIRED.
1 REM  - HOWEVER. AFTER STRING FUNCTIONS HAVE BEEN STUDIED A STU
DENT MIGHT TRY
2 REM  - TO WRITE A CONVERSION PROGRAM.  THIS PROGRAM CONVERTS B
ETWEEN ANY BASES
3 REM  - BETWEEN 2 AND 36. USING ASCII CODES TO GENERATE THE PRO
PER LETTER SYMBOLS.
4 REM  - THE MAX. NO. OF CHARACTERS/NUMBER IS SET AT 15. BUT CAN
 BE CHANGED IN LINE 30.
5 CLEAR 100
7 PRINT TAB( 10);"BASE B1 TO B2 CONVERSION"
8 PRINT :PRINT :PRINT "ENTER * TO REVERSE ORDER":PRINT
9 PRINT :PRINT
10 INPUT "BASE B1 TO B2: ENTER B1,B2:  ";B1,B2
20 IF  B1= B2 OR  B1< > INT (B1) OR  B2< > INT (B2) OR  B1< 2 OR
  B2< 2 THEN 10
30 DIM H(15).HN(15),D(15).R(15)
100 PRINT :PRINT :PRINT "ENTER BASE";B1;" INTEGER: ";:INPUT "";H
$
105 IF H$= "*"THEN C= B1:B1= B2:B2= C:GOTO  100
110 L= LEN (H$)
120 FOR I= 0TO L- 1
130 H(I)= ASC (MID$ (H$.L- I.1))
140 IF (H(I)< 58AND H(I)> 47)OR (H(I)< 91AND H(I)> 64)THEN 160
150 PRINT "ONLY LEGAL NUMBERS. PLEASE!":GOTO 100
160 IF H(I)< 58THEN HN(I)= H(I)- 48
162 IF H(I)> 64THEN HN(I)= H(I)- 55
164 IF HN(I)< 0OR HN(I)> B1- 1THEN 150
166 NEXT I
170 D= 0
180 FOR J= 0TO L- 1
190 MR= INT (B1↑ J+ .1):REM  ROUNDOFF IN B1↑J RAISES HAVOC WITH
R(0)
200 D= D+ MR* HN(J)
210 NEXT J
250 IF D< 1E6THEN 300
260 D1= INT (D/ 1E6):D2= D- 1E6* D1
265 D2$= MID$ (STR$ (D2).2)
267 IF LEN (D2$)< 6THEN D2$= "0"+ D2$:GOTO 267
270 D$= MID$ (STR$ (D1).2)+ D2$
275 PRINT "WARNING:  POSSIBLE ROUNDOFF ERROR"
280 GOTO 305
300 D$= MID$ (STR$ (D).2)
305 IF B2= 10THEN H2$= D$:GOTO 420
310 FOR I= 1TO LEN (D$):M= ASC (MID$ (D$,1.1))
340 D(0)= D:I= 1
350 IF D(I- 1)< B2THEN R(I- 1)= D(I- 1):GOTO 380
360 D(I)= INT (D(I- 1)/ B2):R(I- 1)= D(I- 1)- B2* D(I)
370 I= I+ 1:GOTO 350
380 H2$= ""
385 FOF J= I- 1TO 0STEP - 1
390 IF R(J)< 10THEN H2$= H2$+ RIGHT$ (STR$ (R(J)).1)
400 IF R(J)> = 10THEN H2$= H2$+ CHR$ (R(J)+ 55)
410 R(J)= 0:NEXT J
420 PRINT H$;" (BASE";B1;") = ";H2$;" (BASE";B2;")
430 GOTO 100
```

3. B6 - Matrix multiplication without MAT multiplication.
   This program is somewhat more general than the one under Topic 13 of this guide. It multiplies any two matrices input or given in data statements, if multiplication is possible, and prints both the original matrices and their product matrix.

```
0 REM  - COMPUTER MATH B. OBJECTIVE 6
1 REM  - THIS PROGRAM ILLUSTRATES MATRIX MULTIPLICATION WITHOUT
2 REM  - USE OF MAT COMMANDS.  IT IS RATHER FLEXIBLE. AND MORE
3 REM  - ELABORATE THAN A STUDENT SHOULD BE EXPECTED TO WRITE.
10 INPUT "DIMENSIONS OF 1ST MATRIX (    X    ) : ";N.K
20 INPUT "DIMENSIONS OF 2ND MATRIX (    X    ) : ";L,M
30 IF K< > LTHEN PRINT "ERROR IN DIMENSIONS":GOTO 10
35 DIM A(N.K).B(K.M).C(N.M)
37 INPUT "INPUT OR READ DATA? ";I$
38 IF LEFT$ (I$.1)= "I"THEN L= 1
39 IF LEFT$ (I$.1)= "R"THEN L= 2
40 IF (L- 1)* (L- 2)< > 0THEN 37
45 REM   ENTER A   MATRIX ELEMENTS ROW BY ROW IN 50
50 DATA 1..5..3333333
51 DATA .5..3333333..25
52 DATA .3333333..25..2
60 REM   ENTER B  MATRIX ELEMENTS ROW BY ROW IN 70
70 DATA 9.-36.30
71 DATA -36.192.-180
72 DATA 30.-180.180
77 PRINT :PRINT
80 FOR I= 1TO N
90 FOR J= 1TO K
95 ON LGOTO 105.100
100 READ A(I.J):NEXT J.I
101 GOTO 110
105 PRINT "A(";I;".";J;")";:INPUT A(I,J):NEXT J.I:PRINT
110 FOR I= 1TO K
120 FOR J= 1TO M
125 ON LGOTO 135.130
130 READ B(I.J):NEXT J,I
131 GOTO 140
135 PRINT "B(";I;".";J;")";:INPUT B(I,J):NEXT J,I:PRINT
140 PRINT "A";
145 FOR I= 1TO N
150 FOR J= 1TO K
160 PRINT TAB( 10* J);A(I,J).:NEXT J:PRINT :NEXT I:PRINT :PRINT
165 PRINT "B";
170 FOR I= 1TO K
190 FOR J= 1TO M
200 PRINT TAB( 10* J);B(I.J).:NEXT J:PRINT :NEXT I:PRINT :PRINT
205 PRINT "C=A*B";
210 FOR I= 1TO N
220 FOR J= 1TO M
222 C(I.J)= 0
225 FOR L= 1TO K
230 C(I.J)= C(I.J)+ A(I.L)* B(L.J)
240 NEXT L
250 PRINT TAB( 8+ 14* (J- 1));C(I,J).:NEXT J:PRINT :NEXT I:PRINT
 :PRINT :PRINT

READY
```

4. B6 a- Value of a rank-N determinant.
Supplement B6b lists a simple determinant program which can be generalized to the N x N case. However it cannot handle the calculation if zeros appear on the diagonal of the determinant during the reduction process. The present program corrects this defect.

```
LIST


0 PRINT :PRINT :PRINT :REM   ENTER DATA IN #300, ROW BY ROW
3 INPUT "RANK OF DETERMINANT?  ";N
4 PRINT
5 DIM X(N).A(N.N)
7 REM  - TRIANGULAR REDUCTION METHOD IS USED.  IF DURING THIS
8 REM  - PROCESS A ZERO PIVOT ELEMENT IS ENCOUNTERED, ROWS ARE
9 REM  - INTERCHANGED TO GIVE A NON-ZERO PIVOT.  WHEN THIS
10 REM  - HAPPENS THE COMMENT '0 ON DIAGONAL' IS GIVEN.
11 REM  - STUDENTS WRITING PROGRAMS ARE APT TO OVERLOOK THIS
12 REM  - PROBLEM.
15 FOR R= 1TO N
20 FOR C= 1TO N
30 READ A(R.C)
35 PRINT A(R.C),
40 NEXT C:PRINT :NEXT R
45 PRINT :PRINT
47 D= 1
50 FOR K= 1TO N- 1
55 IF A(K.K)= 0THEN 200
60 FOR R= K+ 1TO N
70 FOR C= NTO KSTEP - 1
75 REM   LINE 70 CHANGES A(R.K) LAST TO AVOID ERROR IN 80
80 A(R.C)= A(R.C)- A(R.K)/ A(K,K)* A(K.C)
90 NEXT C.R
95 NEXT K
110 FOR R= 1TO N
120 D= D* A(R.R)
130 NEXT R
140 PRINT "DETERMINANT = ";D
150 END
200 REM   REMOVE 0 FROM DIAGONAL
203 PRINT "0 ON DIAGONAL"
205 MX= 0:KP= 0
210 FOR R= K+ 1TO N
220 IF ABS (A(R.K))) MXTHEN MX= ABS (A(R.K)):KP= R
230 NEXT R
240 IF MX= 0THEN 95
250 REM   EXCHANGE ROWS
260 FOR C= 1TO N
270 X(C)= A(K.C):A(K.C)= A(KP.C):A(KP.C)= X(C)
280 NEXT C
285 D= - D
290 GOTO 60
300 DATA 0,-2.0.0,-1,5,2,6.4,35,-3,3,0,18.1,-4

READY
```

5. B6a* - Matrix inversion. (The program listing is on the following two pages.)

Supplement B6a shows how to calculate the inverse of a 2 x 2 or a 3 x 3 matrix. This program, based on a Gaussian reduction method, evaluates the inverse of an arbitrary N x N matrix. A problem arises concerning the question of when a matrix is singular. Because of roundoff errors, the determinant of a singular matrix may not appear to be precisely zero. For this reason, lines 2080 and 2620 specify a lower limit, below which the determinant is assumed to be zero. If the program determines that a matrix is singular it still prints a matrix which it labels the inverse. This is a defect in the program. The quoted matrix should be ignored, as a singular matrix has no inverse. This program, more complex than a student could be expected to write, is included for use in solving systems of linear equations, as required for Objective B17(c). For the latter use the program should be modified as suggested in this guide under Additional Objectives B17(c).

5. B6a*- Matrix inversion. (page 2 of 3)

```
LIST


1899 REM   SUBROUTINE "MATINV".  #'S 1910.1915 SET UP MATRIX.
1900 PRINT TAB( 20);"MATRIX INVERSION":PRINT
1901 REM   FINDS DET AND INVERSE MATRIX OF A NON-SINGULAR MATRIX
1902 REM   THE MATRIX ELEMENTS MUST BE ENTERED COLUMN BY COLUMN.
1903 REM   THE PROGRAM ALSO SOLVES A SYSTEM OF LINEAR EQUATIONS.
1904 REM   THE INHOMOGENEOUS TERMS ARE ENTERED IN THE N+1 COLUMN
1905 REM   AND THE VARIABLES ARE RETURNED IN THE SAME COLUMN
1906 REM   THE VALUE OF 'TERM' IS LINE 2080 SETS THE SIZE AT
1907 REM   WHICH THE MATRIX IS CONSIDERED TO BE SINGULAR
1908 REM   NOTE: CERTAIN VARIABLE NAMES MAY NEED TO BE SHORTENED
 TO ONE
1909 REM   OR TWO LETTERS IF THE PROGRAM IS TO WORK ON YOUR SYST
EM.
1910 DATA 1.2,3:REM  - 1ST COLUMN
1911 DATA 2.3,5:REM  - 2ND COLUMN
1912 DATA 3.5.7:REM  - 3RD COLUMN
1913 DATA 0.1.2:REM  - INHOMOGENEOUS TERMS.
1914 REM  - THUS. X + 2Y + 3Z = 0 , ...
1915 N= 3
1920 DIM A(N+ 1,N+ 1)
1950 FOR J= 1TO N+ 1:FOR I= 1TO N
1955 READ A(I.J)
1956 NEXT I.J
1960 PRINT :PRINT :PRINT "ORIGINAL MATRIX";TAB( 15* N);"INHOM. T
ERMS"
1964 FOR I= 1TO N
1965 FOR J= 1TO N+ 1:PRINT TAB( 15* J- 15);A(I,J);:NEXT J
1970 PRINT :NEXT I
1975 GOSUB 2000
1977 PRINT :PRINT :PRINT "INVERSE MATRIX:  DET =";DET;TAB( 15* N
);"VARIABLES"
1980 FOR I= 1TO N:FOR J= 1TO N+ 1
1985 PRINT TAB( 15* J- 15);A(I,J);:NEXT J
1988 PRINT :NEXT I
1990 PRINT :PRINT "NOTE THAT FOR THE GIVEN DATA, THE 3ND ROW OF
THE INVERSE"
1991 PRINT "MATRIX IS 1.1,-1  AND THE SYSTEM OF EQUATIONS HAS SO
LUTION"
1992 PRINT "X=3. Y=0. Z=-1"
1995 END
2000 REM  SUBROUTINE MATINV(A.N,IPVT,NDX,DET)
2005 REM  MATRIX INVERSION WITH ACCOMPANYING SOLUTION OF LINEAR
EQUATIONS
2010 REM  A INVERSE IS RETURNED IN A. N IS MATRIX SIZE.
2015 REM  DETERMINANT IS DET.
2020 DIM IPVT(N),NDX(N,2)
2040 DET= 1
2060 NP1= N+ 1
2080 TERM= 1E- 5
2100 FOR J= 1TO N
2120 IPVT(J)= 0:NEXT J
2140 FOR I= 1TO N:REM   TO 2840
```

5. B6a*- Matrix inversion. (page 3 of 3)

```
2145 REM  SEARCH FOR PIVOT ELEMENT
2160 AMAX= 0
2180 FOR J= 1TO N:REM  TO 2380
2200 IF IPVT(J)= 1THEN 2380
2220 FOR K= 1TO N
2240 IF IVPT(K)> 1THEN PRINT "MATRIX ERROR":RETURN
2260 IF IPVT(K)= 1THEN 2360
2280 IF ABS (AMAX)> ABS (A(J.K))THEN 2360
2300 IROW= J
2320 ICLM= K
2340 AMAX= A(J.K)
2360 NEXT K
2380 NEXT J
2400 IPVT(ICLM)= IPVT(ICLM)+ 1
2405 REM  INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
2420 IF IROW= ICLMTHEN 2540
2440 DET= - DET
2460 FOR L= 1TO NP1
2480 SWAP= A(IROW.L)
2500 A(IROW.L)= A(ICLM.L)
2520 A(ICLM.L)= SWAP:NEXT L
2540 NDX(I.1)= IROW
2560 NDX(I.2)= ICLM
2580 PVT= A(ICLM.ICLM)
2600 DET= DET* PVT
2620 IF ABS (DET)< TERMTHEN PRINT "SINGULAR MATRIX":RETURN
2625 REM  DIVIDE PIVOT ROW BY PIVOT ELEMENT
2640 A(ICLM.ICLM)= 1
2660 FOR L= 1TO NP1
2680 A(ICLM.L)= A(ICLM.L)/ PVT:NEXT L
2685 REM   REDUCE NON-PIVOT ROWS
2700 FOR L1= 1TO N
2720 IF L1= ICLMTHEN 2820
2740 T= A(L1.ICLM)
2760 A(L1.ICLM)= 0
2780 FOR L= 1TO NP1
2800 A(L1.L)= A(L1.L)- A(ICLM.L)* T:NEXT L
2820 NEXT L1
2840 NEXT I
2850 REM  INTERCHANGE COLUMNS
2860 FOR I= 1TO N
2880 L= N+ 1- I
2900 IF NDX(L.1)= NDX(L.2)THEN 3060
2920 JROW= NDX(L.1)
2940 JCLM= NDX(L.2)
2960 FOR K= 1TO N
2980 SWAP= A(K.JROW)
3000 A(K.JROW)= A(K.JCLM)
3020 A(K.JCLM)= SWAP
3040 NEXT K
3060 NEXT I
3080 RETURN
3100 END

READY
```

6. B9a - Compucolor Files.
This guide discusses Microsoft/TRSDOS files under Additional Objectives B9a. This program gives examples of files as used by Intelligent Systems Corporation computers, and in particular, by the Compucolor II.

```
0 CLEAR 500
1 REM   THE PROPER USE OF FILES IS STRONGLY SYSTEM DEPENDENT.
2 REM   THIS PROGRAM ILLUSTRATES USE OF RANDOM FILES
3 REM   ON THE COMPUCOLOR II.
4 REM   INFORMATION IS STORED IN A RANDOM FILE CALLED "INV"
5 REM   THIS PROGRAM OPENS A RANDOM FILE (PREVIOUSLY CREATED
6 REM   AND SAVED ON A FLOPPY DISK). GETS SOME INFORMATION
7 REM   FROM THE FILE. UPDATES THE INFORMATION. "PUTS" THE
8 REM   INFORMATION BACK IN THE FILE. AND CLOSES IT.  IT THEN LO
ADS
9 REM   AND RUNS THE SEPARATE PROGRAM "INPRNT". WHICH PRINTS THE
 UPDATED RESULTS.
10 PLOT 12:DIM TIT$(10).NM$(10)
20 FILE "R".1."INV",1
25 PUT 1.1.89;"REVINV"[6]
30 REM  TITLES ALL IN FILE 1, NO. ON HAND ALL IN FILE 4
40 GET 1.1.1;TIT$(0)[10]:GET 1,4,1;NM$(0)[10]
45 PRINT TIT$(0);SPC( 2);NM$(0):PRINT
50 FOR I= 1TO 10
60 GET 1.1.6* I+ 5;TIT$(I)[6]:GET 1.4.6* I+ 5;NM$(I)[6]
70 PRINT TIT$(I);TAB( 12);NM$(I):NEXT I
80 PRINT :PRINT "ENTER NAME OF ITEM WHOSE INVENTORY YOU WISH TO
UPDATE "
90 PRINT "TO END THE LIST TYPE 'NONE'"
100 INPUT A$
110 IF A$= "NONE"THEN 200
120 FOR I= 1TO 10
130 IF A$= LEFT$ (TIT$(I).3)THEN 150
140 NEXT I:GOTO 80
150 INPUT "NO. SOLD: ";NS
160 INPUT "NO. RESTOCKED: ";NR
170 GOTO 250:REM  FOR STRING CONVERSION
200 FILE "C".1
210 LOAD "INPRNT":RUN
250 N= VAL (NM$(I))+ NR- NS
260 IF N< 0THEN PRINT "YOU SOLD MORE THAN YOU HAVE!":GOTO 150
270 PRINT TIT$(I).N
280 NM$(I)= MID$ (STR$ (N).2)
290 IF LEN (NM$(I))< 3THEN NM$(I)= " "+ NM$(I):GOTO 290
300 PUT 1.4,6* I+ 5;NM$(I)[6]
310 GOTO 80
```

THESE TWO PROGRAMS ILLUSTRATE THE USE OF A TYPE OF DATA FILE. CALLED AN ARRAY FILE. AVAILABLE ON COMPUCOLOR II. TWO INDEPENDENT PROGRAMS ARE ACTUALLY INCLUDED. THE FIRST SAVES A SET OF 12 NUMBERS ON A FLOPPY DISK; THE SECOND LOADS THE NUMBERS FROM DISK AND PRINTS THEM.

```
10 REM  SAVE MONTHS
20 DIM M(12)
30 FOR I=1 TO 12
40 READ M(I)
50 NEXT I
60 DATA 31,28.31.30,31.30
70 DATA 31.31.30.31.30.31
80 SAVE"M.ARY"
```

```
10 REM  LOAD MONTHS
20 DIM M(12)
30 LOAD"M.ARY"
40 FOR I=1 TO 12
50 PRINT M(I).
60 NEXT I
```

USE OF FILES ALLOWS MORE THAN ONE BASIC PROGRAM TO SHARE THE SAME INFORMATION.

7. B13a - PEEK and POKE.

This program illustrates PEEK and POKE as used on the Compucolor II. Use on other computer systems should be similar.

```
0 REM  - THIS PROGRAM ILLUSTRATES PEEK AND POKE FEATURES.
1 REM  - IN PARTICULAR. IT USES THE REAL-TIME CLOCK OF THE
2 REM  - COMPUCOLOR II MICROCOMPUTER.  HOWEVER, SIMILAR
3 REM  - CONSIDERATIONS USING THE PEEK FUNCTION AND THE
4 REM  - POKE COMMAND MAY APPLY TO OTHER SYSTEMS IN USE.
5 REM  - MEMORY LOCATION 33278 STORES THE ASCII CODE FOR THE
6 REM  - KEYBOARD CHARACTER MOST RECENTLY TOUCHED.  THUS IN
7 REM  - LINE 35 THE PROGRAM RESPONDS TO A 'Y' OR 'N' WITHOUT
8 REM  - THE NEED FOR PUSHING THE 'RETURN' KEY.
9 REM  - LINES 18 AND 30 HAD PREVIOUSLY PUT THE COMPUTER ON A
10 REM  - WAIT STATUS.
11 REM  - HOURS, MINUTES, AND SECONDS ARE STORED, RESPECTIVELY,
12 REM  - IN LOCATIONS 33211,33210,33209.  THE CLOCK IS SET BY
13 REM  - POKING IN DESIRED VALUES, AND READ BY PEEKING AT THEM.
14 REM  - CURRENT TIME IS DISPLAYED IN THE UPPER RIGHT-HAND
15 REM  - CORNER OF THE SCREEN (LOCATED BY THE PLOT STATEMENT OF
 LINE 150)
16 REM  - OTHER PLOT STATEMENTS CLEAR THE SCREEN AND CHOOSE LETT
ER SIZE AND COLOR.
17 PLOT 12.15
18 POKE 33278.0
20 PRINT "DO YOU WISH TO SET THE CLOCK? "
25 PLOT 6.0
30 IF PEEK (33278)= 0THEN 30
35 IF PEEK (33278)< > ASC ("Y")THEN 90
38 PRINT :PLOT 6.2
40 INPUT "TYPE H.M.S: ";H.M.S
50 POKE 33209,S:POKE 33210,M:POKE 33211.H
90 PLOT 12.6.2
100 S= PEEK (33209)
110 IF S= PEEK (33209)THEN 110
120 M= PEEK (33210):H= PEEK (33211):S= PEEK (33209)
130 M$= RIGHT$ (STR$ (100+ M),2):S$= RIGHT$ (STR$ (100+ S),2)
140 H$= RIGHT$ (STR$ (100+ H),2)
150 PLOT 3.50.0:PRINT H$;":";M$;":";S$
160 GOTO 100

READY

LINES 130 AND 140, INCIDENTALLY, ALLOW INITIAL ZEROES TO BE PRINTED. FOR EXAMPLE

04:30:08  CAN BE SHOWN AS THE CURRENT TIME..
```

8. B14 - String functions.
Supplements B14.1 and B14.2 discuss string functions and give some exercises for the student. The present program gives further examples.

```
0 REM   SOME EXAMPLES OF THE USE OF STRING FUNCTIONS
10 A$= "APPLESAUCE":B$= "CRACKERJACK"
20 PRINT LEFT$ (A$.5)
30 PRINT MID$ (A$.6)
40 PRINT LEFT$ (A$.5)+ RIGHT$ (B$.4)
50 PRINT MID$ (B$.1.7)+ MID$ (A$.6.5)
60 PRINT MID$ (A$.1.6)+ MID$ (B$.9)
70 PRINT CHR$ (ASC (B$))
80 PRINT VAL ("3.14")
90 PRINT VAL (A$)
100 C$= STR$ (3.14)
110 PRINT LEN (C$)
120 PRINT ASC (LEFT$ (C$.1)): REM   THE FIRST CHARACTER OF C$ IS
 A BLANK
130 PRINT RIGHT$ (STR$ (1000+ 85),3)

READY

RUN

APPLE
SAUCE
APPLEJACK
CRACKERSAUCE
APPLESACK
C
 3.14
 0
 5
 32
085
```

9. B17a(a) - Addition and subtraction of fractions.
String functions can be used to determine the particular format in which information is entered. This program will add or subtract fractions and reduce the result to lowest terms (using the Euclidean algorithm for finding the gcf), when the information is entered in normal algebraic form, for example, 2/3 - 3/7 . A typical run is shown below. The program is listed on the following page.

```
RUN

THIS PROGRAM ADDS OR SUBTRACTS FRACTIONS.
ENTER IN NORMAL FORM. FOR EXAMPLE. 1/2 - 1/6


?2/3 - 3/7


2/3 - 3/7 = 5 / 21


?2/3 + 3/4


2/3 + 3/4 = 17 / 12
```

9. B17a(a) - Addition and subtraction of fractions.
Program listing. Output and description are on the previous page.

```
LIST


0 REM  COMP. MATH. B. OBJECTIVE 17A(A).
1 REM   STRING FUNCTIONS ARE USED TO ALLOW ENTERING THE SUM OR
2 REM   DIFFERENCE OF FRACTIONS IN NORMAL ALGEBRAIC FORM.
3 REM   THE EUCLIDEAN ALGORITHM FOR THE GCF OF TWO NUMBERS IS
4 REM   USED IN THE SUBROUTINE BEGINNING ON LINE 700.
10 PRINT "THIS PROGRAM ADDS OR SUBTRACTS FRACTIONS."
20 PRINT "ENTER IN NORMAL FORM, FOR EXAMPLE, 1/2 - 1/6"
30 PRINT :PRINT
90 DIM B(15)
100 REM  DETERMINE NUMERATOR AND DENOMINATIOR OF FRACTIONS
110 INPUT A$:REM   A$ OF FORM  A/B + C/D
120 N1$= "":C= LEN (A$)
130 FOR I= 1TO C
140 B$= MID$ (A$,I,1)
150 IF B$= "/"THEN 210
160 N1$= N1$+ B$
170 NEXT I
210 N1= VAL (N1$)
220 D1$= ""
230 FOR J= I+ 1TO C
240 B$= MID$ (A$,J,1)
250 IF (B$= "+"OR B$= "-")THEN 310
260 D1$= D1$+ B$
270 NEXT J
310 D1= VAL (D1$)
320 N2$= B$
330 FOR K= J+ 1TO C
340 B$= MID$ (A$,K,1)
350 IF B$= "/"THEN 410
360 N2$= N2$+ B$
370 NEXT K
410 N2= VAL (N2$)
420 D2$= ""
430 FOR L= K+ 1TO C
440 B$= MID$ (A$,L,1)
460 D2$= D2$+ B$
470 NEXT L
480 D2= VAL (D2$)
500 REM   ALGEBRA
520 NU= N1* D2+ N2* D1
530 DE= D1* D2
540 GOSUB 700
550 REM  GCF OF NU AND DE
560 NU= NU/ GCF
570 DE= DE/ GCF
580 PRINT :PRINT A$;" =";NU;" /";DE
590 PRINT :PRINT :GOTO 100
700 REM  GCF BY EUCLIDEAN ALGORITHM
710 B(0)= NU* SGN (NU):B(1)= DE
720 IF B(1)> B(0)THEN B= B(0):B(0)= B(1):B(1)= B
730 I= 0
740 Q= INT (B(I)/ B(I+ 1)):B(I+ 2)= B(I)- Q* B(I+ 1)
750 IF B(I+ 2)< > 0THEN 770
760 GCF= B(I+ 1)
765 RETURN
770 I= I+ 1:GOTO 740
```

10. B17a(b) - Cube roots of a real number.
Every real number has three real cube roots, two of which are complex. This program calculates all three roots for any given real number.

```
LIST


10 REM   CUBE ROOTS OF A REAL NUMBER
20 REM   THE THREE ROOTS OF N ARE R=N↑(1/3), R COS(A) + IR SIN(A
),
30 REM   AND R COS(A) - IR  SIN(A), WHERE I↑2=-1 AND A=2*PI/3
40 REM   A IS GIVEN IN RADIANS, SINCE THAT IS WHAT THE COMPUTER
50 REM   USES.
60 REM   IN PARTICULAR, COS(A) = -0.5 AND SIN(A) = SQR(3)/2 .
100 A= 2* 3.141593/ 3
110 INPUT "NUMBER WHOSE CUBE ROOTS ARE TO BE FOUND: ";N
120 R= SGN (N)* ABS (N)↑ (1/ 3):REM   THE COMPUTER NEEDS A POSIT
IVE BASE
130 R1= - R/ 2:R2= R* SQR (3)/ 2
135 PRINT "THE ROOTS ARE"
140 PRINT R
150 PRINT R1;" + I (";R2;")"
160 PRINT R1;" - I (";R2;")"

READY

RUN

NUMBER WHOSE CUBE ROOTS ARE TO BE FOUND: 8

THE ROOTS ARE
 2
-1 + I ( 1.73205)
-1 - I ( 1.73205)


READY

RUN

NUMBER WHOSE CUBE ROOTS ARE TO BE FOUND: -27

THE ROOTS ARE
-3
 1.5 + I (-2.59808)
 1.5 - I (-2.59808)
```

11. B17a(c).1 - Multiple-precision addition.
This program, with listing on the next page and a typical run on the page following that, uses string techniques to add two or more integers of arbitrary length. It could easily be generalized to allow addition of decimal numbers.

BEST COPY AVAILABLE

11. B17a(c).1 - Multiple-precision addition. (page 2 of 3)

```
1 REM   THIS PROGRAM ADDS TWO OR MORE POSITIVE INTEGERS OF
2 REM   ARBITRARY LENGTH.  GENERALIZATION TO INCLUDE PROPER
3 REM   TREATMENT OF DECIMAL POINTS IS LEFT TO THE INGENUITY
4 REM   OF THE INTERESTED STUDENT OR TEACHER.
10 CLEAR 500
20 PRINT "PLEASE ENTER ONLY POSITIVE INTEGERS (NO COMMAS)"
30 INPUT "NUMBER OF INTEGERS TO BE ADDED:  ";N
40 IF N< 2OR N< > INT (N)THEN 30
45 IF N> 10THEN DIM A$(N),L(N),B$(N)
50 FOR I= 1TO N
60 PRINT "INTEGER NO.";I;:INPUT ":  ";A$(I)
70 L(I)= LEN (A$(I)):NEXT I
75 REM  REMOVE BLANKS AND CHECK FOR NUMERALS
80 FOR K= 1TO N:B$(K)=""
90 FOR J= 1TO L(K)
100 T$= MID$ (A$(K),J,1)
110 IF T$= " "THEN 135
120 IF ASC (T$)< 48OR ASC (T$)> 57THEN 20
130 B$(K)= B$(K)+ T$
135 NEXT J
140 L(K)= LEN (B$(K)):NEXT K
145 REM  FIND MAXIMUM LENGTH
150 L= 1
160 FOR K= 2TO N
170 IF L(K)> L(L)THEN L= K:NEXT K
175 REM   LEFT EXTEND ALL NUMBERS TO THE SAME LENGTH
180 FOR K= 1TO N
190 IF L(K)< L(L)THEN B$(K)= "0"+ B$(K):L(K)= L(K)+ 1:GOTO 190
200 NEXT K
205 REM   ADD DIGITS
210 C= 0:DIM S(L(L)),S$(L(L))
220 FOR M= L(L)TO 1STEP - 1
230 S(M)= 0
240 FOR I= 1TO N
250 S(M)= S(M)+ VAL (MID$ (B$(I),M,1))
260 NEXT I:S(M)= S(M)+ C
270 C= INT (S(M)/ 10):S(M)= S(M)- 10* C
280 S$(M)= RIGHT$ (STR$ (S(M)),1)
290 NEXT M
300 S(0)= C:S$(0)= MID$ (STR$ (C),2)
305 REM   PRINT OUT RESULTS, GROUPED BY SIXES
310 PRINT :PRINT
320 FOR I= 1TO N:FZ= 0
330 IF I= NTHEN PRINT "+";
340 FOR J= 1TO L(L)
350 PRINT TAB( 3)
360 T$= MID$ (B$(I),J,1)
370 IF FZ= 0THEN FZ= FZ+ VAL (T$)
380 IF INT ((L(L)- J+ 1)/ 6)* 6= L(L)- J+ 1THEN PRINT " ";
390 IF FZ= 0THEN PRINT " ";:GOTO 410
400 PRINT T$;
410 NEXT J:PRINT :NEXT I
420 PRINT TAB( 2)
430 FOR J= 1TO L(L)+ 1+ INT (L(L)/ 5)
440 PRINT CHR$ (45);:NEXT J:PRINT
450 PRINT "=";
460 PRINT TAB( 3- LEN (S$(0)))
470 IF S$(0)= "0"THEN PRINT " ";:GOTO 490
480 PRINT S$(0);
490 FOR J= 1TO L(L)
495 IF INT ((L(L)- J+ 1)/ 6)* 6= L(L)- J+ 1THEN PRINT " ";
500 PRINT S$(J);:NEXT J : PRINT:PRINT: RUN 30
```

11. B17a(c).1 - Multiple-precision addition. (page 3 of 3)

```
RUN

PLEASE ENTER ONLY POSITIVE INTEGERS (NO COMMAS)
NUMBER OF INTEGERS TO BE ADDED:  4

INTEGER NO. 1:  234345456

INTEGER NO. 2:  97532

INTEGER NO. 3:  2468097532468

INTEGER NO. 4:  999888777666555


   234 345456
        97532
   246 809753
+  999 888777
  -----------
= 1481 141518

NUMBER OF INTEGERS TO BE ADDED:  3

INTEGER NO. 1:  9876543212345678909876543

INTEGER NO. 2:  3570987654321244567

INTEGER NO. 3:  23456788765555588888888887


   9 876543 212345 678909 876543
          3 570987 654321 244567
+  2 345678 876555 558888 888888
  -------------------------------
= 12 222225 659888 892120 009998

NUMBER OF INTEGERS TO BE ADDED:
```

12. B17a(c).2 - Multiple-precision multiplication. (Page 1 of 2)
This program uses string techniques to allow multiplication of two real numbers of arbitrary length. It also has the feature of preserving the product for use in a further chained multiplication.

```
0 REM  - THIS PROGRAM ALLOWS MULTIPLICATION OF TWO REAL NUMBERS
1 REM  - OF ARBITRARY LENGTH.  IT ALSO ALLOWS FOR CHAIN
2 REM  - MULTIPLICATION SO THAT, FOR EXAMPLE, 20! COULD BE
3 REM  - CALCULATED EXACTLY.
4 REM  - SIGN AND DECIMAL POINT ARE PROPERLY INDICATED.
5 REM  - IF A STUDENT TRIES THIS PROJECT HE SHOULD FIRST ATTEMPT
6 REM  - TO WRITE A PROGRAM FOR POSITIVE INTEGERS.
15 CLEAR 500
20 PRINT TAB( 15);"MULTIPLE-PRECISION MULTIPLICATION"
30 PRINT :PRINT
40 INPUT "FIRST NUMBER?  ";A$
50 ST= 1:LA= LEN (A$):TA$= ""
55 DA= 0:SA$= " ":IF LEFT$ (A$,1)= "-"THEN SA$= "-":ST= 2
60 FOR I= STTO LA
70 T= ASC (MID$ (A$,I,1))
80 IF T= 46AND DA< > 0THEN PRINT "TOO MANY DECIMAL POINTS!":GOTO
 40
85 IF T= 46THEN DA= I+ 2- ST
90 IF T= 46OR (T> 47AND T< 58)THEN TA$= TA$+ MID$ (A$,I,1):GOTO
100
95 PRINT "THAT ISN'T A NUMBER.  TRY AGAIN!":GOTO 40
100 NEXT I
105 IF DA= 0THEN DA= LEN (A$)+ 2- ST
110 IF LEFT$ (TA$,1)= "0"THEN TA$= MID$ (TA$,2):DA= DA- 1:GOTO 1
10
120 A$= SA$+ TA$
140 INPUT "SECOND NUMBER? ";B$
150 ST= 1:LB= LEN (B$):TB$= ""
155 DB= 0:SB$= " ":IF LEFT$ (B$,1)= "-"THEN SB$= "-":ST= 2
160 FOR I= STTO LB
170 T= ASC (MID$ (B$,I,1))
180 IF T= 46AND DB< > 0THEN PRINT "TOO MANY DECIMAL POINTS!":GOT
O 140
185 IF T= 46THEN DB= I+ 2- ST
190 IF T= 46OR (T> 47AND T< 58)THEN TB$= TB$+ MID$ (B$,I,1):GOTO
 200
195 PRINT "THAT ISN'T A NUMBER.  TRY AGAIN!":GOTO 140
200 NEXT I
201 IF DB= 0THEN DB= LEN (B$)+ 2- ST
202 IF LEFT$ (TB$,1)= "0"THEN TB$= MID$ (TB$,2):DB= DB- 1:GOTO 2
02
205 B$= SB$+ TB$
210 IF LEN (A$)< LEN (B$)THEN C$= B$:B$= A$:A$= C$:DC= DA:DA= DB
:DB= DC
230 DP= LEN (A$)+ LEN (B$)- DA- DB:REM   LOCATION OF DECIMAL POI
NT
240 SG$= " ":REM   SIGN OF PRODUCT
250 IF ASC (LEFT$ (A$,1))+ ASC (LEFT$ (B$,1))= 77THEN SG$= "-"
260 PRINT :PRINT TAB( LEN (B$));A$
270 PRINT TAB( LEN (A$));B$;:MK= POS (X):PRINT
280 FOR J= 1TO LEN (A$)+ LEN (B$)
290 PRINT  CHR$ (45);:NEXT J:PRINT
300 IF DA< LEN (A$)THEN A$= LEFT$ (A$,DA- 1)+ MID$ (A$,DA+ 1)
305 A$= MID$ (A$,2)
310 IF DB< LEN (B$)THEN B$= LEFT$ (B$,DB- 1)+ MID$ (B$,DB+ 1)
315 B$= MID$ (B$,2)
390 REM   MULTIPLICATION
400 LA= LEN (A$):LB= LEN (B$)
405 P$= "":CA= 0
410 FOR I= LA+ LBTO 1STEP - 1  : P = 0
```

BEST COPY AVAILABLE

12. Bi7a(c).2 - Multiple-precision multiplication.   (page 2 of 2)

```
420 FOR J= 1TO LB
430 IF I- J> LAOR I- J< 1THEN 450
440 P= VAL (MID$ (A$.I- J.1))* VAL (MID$ (B$.J.1))+ P
450 NEXT J
455 P= P+ CA
460 CA= INT (P/ 10):P= P- 10* CA
470 P$= RIGHT$ (STR$ (P).1)+ P$
480 NEXT I
500 IF DP= LA+ LBTHEN P$= "0."+ P$:GOTO 520
510 IF DP> 0THEN P$= LEFT$ (P$.LA+ LB- DP)+ "."+ RIGHT$ (P$.DP)
515 IF LEFT$ (P$.1)= "0"AND MID$ (P$.2.1)< > "."THEN P$= MID$ (P
$.2):GOTO 515
520 P$= SG$+ P$
530 PRINT TAB( MK- LEN (P$));P$
540 PRINT
600 INPUT "CHAIN MULTIPLICATION? ";Z$
610 PRINT :PRINT :PRINT :IF LEFT$ (Z$.1)= "Y"THEN 630
620 GOTO 30
630 IF SG$= " "THEN P$= MID$ (P$.2)
640 A$= P$:GOTO 50

READY

RUN

               MULTIPLE-PRECISION MULTIPLICATION


FIRST NUMBER?  -234.5678932

SECOND NUMBER? 28.76543234567


             28.76543234567
               -234.5678932
---------------------------
   -5747.446862310946042444

CHAIN MULTIPLICATION? Y




SECOND NUMBER? -888.777777


           -5747.446862310946042444
                        -888.777777
-----------------------------------
   5996980.822710347706370325966508

CHAIN MULTIPLICATION? N
```

13. B17a(e) - **Stirling's Approximation to N!**
This program illustrates Stirling's approximation to the factorial function. An improvement to this approximation is mentioned in the discussion under Additional Objective B17a(e) in this guide.

```
LIST


0 REM   COMPUTER MATH B. OBJECTIVE 17a(e):  STIRLING'S APPROXIMA
TION TO N!
5 REM   THE APPROXIMATION IS ACCURATE TO BETTER THAN 1% FOR INTE
GERS LARGER THAN 8.
20 PI= 3.141593
30 DEF FN F(N)= SQR (2* PI* N)* N↑ N* EXP (- N)
40 PRINT "N";TAB( 8);"F(N)";TAB( 24);"F(N)/N!":PRINT
50 FOR N= 1TO 15
60 P= 1:NF= FN F(N)
70 FOR I= 1TO N
80 P= P* I:NEXT I
90 PRINT N;TAB( 8);NF;TAB( 24);NF/ P
100 NEXT N

READY

RUN

N        F(N)            F(N)/N!

 1        .922137         .922137
 2        1.91901         .959503
 3        5.83622         .972702
 4        23.5062         .979424
 5        118.019         .983494
 6        710.078         .98622
 7        4980.4          .988175
 8        39902.4         .989643
 9        359537          .990788
 10       3.5987E+06      .991705
 11       3.96158E+07     .99246
 12       4.75688E+08     .993082
 13       6.18727E+09     .993617
 14       8.66612E+10     .994069
 15       1.30044E+12     .994467

READY
```

14. B17a(f).1 - Newton's method for polynomial.
Newton's method for finding zeros of a function is here illustrated for a third-degree polynomial. The program is identical to one which could be written for the cubic equation portion of Objective B7a. The equation has three distinct roots.

```
LIST


10 REM   NEWTON'S METHOD FOR ZEROS OF A POLYNOMIAL
20 INPUT "INITIAL VALUE OF X IS: ";X
30 IF X= 999THEN END
50 DEF FN F(X)= X↑ 3- 4* X+ 2
60 DEF FN D(X)= 3* X↑ 2- 4
70 REM   D(X) IS THE DERIVATIVE FUNCTION OF F(X)
80 S= X- FN F(X)/ FN D(X)
90 T= ABS (S- X)
100 IF T< 1E- 5THEN 140
110 X= S
120 GOTO 80
140 PRINT "FINAL VALUE OF X IS ";X
150 PRINT
160 GOTO 20
170 END

READY

RUN

INITIAL VALUE OF X IS: -2

FINAL VALUE OF X IS -2.21432

INITIAL VALUE OF X IS: 0

FINAL VALUE OF X IS  .539189

INITIAL VALUE OF X IS: 2

FINAL VALUE OF X IS  1.67513

INITIAL VALUE OF X IS: 999


READY
```

BEST COPY AVAILABLE

15. B17a(f).2 - Newton's method for trigonometric function.
The same program as the previous one is applied to solving the trigonometric equation $x = \cos(x)$. The equation has only one root.

```
LIST


10 REM   NEWTON'S METHOD FOR ZEROS OF A TRIGONOMETRIC FUNCTION
15 REM   SOLVES THE EQUATION  X=COS(X)
20 INPUT "INITIAL VALUE OF X IS: ";X
30 IF X= 999THEN END
50 DEF FN F(X)= X- COS (X)
60 DEF FN D(X)= 1+ SIN (X)
70 REM   D(X) IS THE DERIVATIVE FUNCTION OF F(X)
80 S= X- FN F(X)/ FN D(X)
90 T= ABS (S- X)
100 IF T< 1E- 5THEN 140
110 X= S
120 GOTO 80
140 PRINT "FINAL VALUE OF X IS ";X
150 PRINT
160 GOTO 20
170 END

READY

RUN

INITIAL VALUE OF X IS: 0

FINAL VALUE OF X IS  .739085

INITIAL VALUE OF X IS: 1

FINAL VALUE OF X IS  .739085

INITIAL VALUE OF X IS: 999


READY
```

BEST COPY AVAILABLE

16. B17a(g).1 - Integration by Simpson's rule.

This program uses Simpson's rule (See discussion of Objection B17a(g) in this guide) to approximate the integral of the function $1/(x^2 + x)$ over the interval (1,2). The approximation improves as the number of intervals chosen increases. Since a definite integral can be interpreted as an area under a curve, once can alternatively, regard this program as one for finding the area under a curve.

```
LIST


10 REM   AREA UNDER A CURVE BY SIMPSON'S RULE
20 DEF FN F(X)= 1/ (X↑ 2+ X):REM  - FUNCTION TO BE INTEGRATED
30 A= 1:REM  - LEFT ENDPOINT
40 B= 2:REM  - RIGHT ENDPOINT
50 INPUT "NUMBER OF INTERVALS: ";N
55 IF N/ 2) > INT (N/ 2)THEN 50
56 IF N= 0THEN END
60 H= (B- A)/ N:REM  - INTERVAL SIZE
70 S= FN F(A)+ FN F(B)
80 FOR I= 2TO N- 2STEP 2
90 S= S+ 2* FN F(A+ I* H)
100 NEXT I
112 FOR J= 1TO N- 1STEP 2
120 S= S+ 4* FN F(A+ J* H)
130 NEXT J
140 R= H* S/ 3
145 PRINT
150 PRINT "IF NO. OF INTERVALS IS ";N;" ,THE APPROX. AREA IS";R
160 PRINT :GOTO 50

READY

RUN

NUMBER OF INTERVALS: 4


IF NO. OF INTERVALS IS  4 .THE APPROX. AREA IS .287783

NUMBER OF INTERVALS: 20


IF NO. OF INTERVALS IS  20 .THE APPROX. AREA IS .287682

NUMBER OF INTERVALS: 40


IF NO. OF INTERVALS IS  40 .THE APPROX. AREA IS .287682

NUMBER OF INTERVALS:


READY
```

BEST COPY AVAILABLE

17. B17a(g).2 - BASIC subroutine, Simpson's rule.
The Simpson's rule program is here written in the form of a subroutine. The function F(X), the interval endpoints X1, X2, and the number N of intervals chosen should be specified in the portion of the program that utilizes the subroutine. The logic of this program is somewhat more compact than that of the previous one.

```
1010 REM   SUBROUTINE SIMP
1020 REM   X1 AND X2 ARE ENDPOINTS OF INTERVAL
1030 REM   N (EVEN) IS THE NUMBER OF SUBINTERVALS IN (X1,X2)
1040 REM   FN F(X) IS THE FUNCTION TO BE INTEGRATED
1050 REM   A1 IS THE APPROXIMATE AREA
1070 A1= FN F(X1)- FN F(X2)
1080 D= (X2- X1)/ N
1090 A= 1
1100 X= X1+ D
1110 FOR L= 1TO N
1120 A1= A1+ (3+ A)* FN F(X)
1130 X= X+ D
1140 A= - A
1150 NEXT L
1160 A1= A1* D/ 3
1170 RETURN
```

18. B17a(g).3 - FORTRAN subroutine, Simpson's rule.
A FORTRAN version of the Simpson's rule subroutine shows a striking similarity to the BASIC version above. Values of XL, XR, N and ANS are conveyed to the calling program as arguments of the subroutine. The function to be integrated needs to be defined by a function statement or function subprogram.

```
EXAMPLE OF A FORTRAN SUBROUTINE:  SIMPSON'S RULE


      SUBROUTINE SIMP(XL.XR,N.ANS)
C     XL AND XR ARE ENDPOINTS OF INTERVAL
C     N (EVEN) IS THE NUMBER OF SUBINTERVALS IN (XL.XR)
C     FN(X) IS THE FUNCTION TO BE INTEGRATED
C     ANS IS THE APPROXIMATE AREA
      ANS = FN(XL) - FN(XR)
      DEL = (XR - XL) /  FLOAT(N)
      A = 1.
      X = XL + DEL
      DO 1 L = 1.N
      ANS = ANS + (3. + A) * FN(X)
      X = X + DEL
    1 A = -A
      ANS = ANS * DEL / 3.
      RETURN
      END
```

BEST COPY AVAILABLE

19. B17a(1).1 - Prime numbers, sieve method.
This program calculates primes from 2 up to N, where the largest allowed N value depends upon available memory. It is much faster than the programs A17(c), requiring less than 4 seconds to calculate primes to 300 on the Compucolor II, as compared to 20 seconds for the fastest of the previous programs.

```
 0 REM  - PRIMES BY THE METHOD OF THE SIEVE OF ERATOSTHENES
 10 INPUT "LIST OF PRIMES UP TO WHAT NUMBER? ";N
 20 DIM A(N)
 30 FOR I= 3TO NSTEP 2
 40 A(I)= I
 50 NEXT I
 60 FOR J= 3TO SQR (N)+ 1E- 5STEP 2
 65 REM  - BEC. OF ROUNDOFF, SQR(I*I) CAN BE LESS THAN I
 70 IF A(J)= 0THEN 100
 75 REM  - IN 70. REJECT NUMBERS FOUND NOT PRIME
 80 FOR K= J* JTO NSTEP 2* J
 90 A(K)= 0:NEXT K
 100 NEXT J
 110 PRINT 2.
 120 FOR I= 3TO NSTEP 2
 130 IF A(I)= 0THEN 150
 140 PRINT A(I).
 150 NEXT I
```

20. B17a(1).2 - Prime numbers, sieve method - extended range.
This program extends the previous one to larger primes, by allowing calculation of primes between N1 and N2. With this program, slightly slower than the previous one, primes as large as 1000000 can be easily calculated. For example, the eight primes between 999900 and 1000000 are determined in 24 seconds on the Compucolor II.

```
2 REM   CALCULATES PRIME NUMBERS BETWEEN N1 AND N2
5 INPUT "TYPE N1 AND N2 (EVEN. >=2)    ";N1,N2
10 DIM A(N2- N1)
30 FOR I= N1+ 1TO N2STEP 2
40 A(I- N1)= I
50 NEXT
60 FOR J= 3TO SQR (N2)+ 1E- 5STEP 2
65 IF J< N1THEN 70
66 IF A(J- N1)= 0THEN 90
68 FOR K= J* JTO N2STEP 2* J
69 GOTO 92
70 C= J+ J* INT (N1/ J)
72 IF C= 2* INT (C/ 2)THEN C= C+ J
75 FOR K= CTO N2STEP 2* J
80 IF K> N2THEN 90
82 A(K- N1)= 0
85 NEXT K
90 NEXT J
95 IF N1< = 3THEN PRINT 2.
100 FOR I= N1+ 1TO N2STEP 2
110 IF A(I- N1)= 0THEN 130
120 PRINT A(I- N1).
130 NEXT I
```

BEST COPY AVAILABLE

21. B17a(j).1 - Shellsort sorting program.
This is a Shellsort routine, modified for alphabetization. The method is explained in Supplement B17a(j).

```
LIST


10 REM  PROGRAM FOR ALPHABETIZING A LIST OF N WORDS
15 REM   THE VALUE OF N SHOULD BE THE FIRST ITEM OF DATA.
LINE 200
20 REM   THE SORTING PROCEDURE USED IS CALLED SHELLSORT
40 CLEAR 1000:READ N
50 DIM A$(N)
60 FOR I= 1TO N
70 READ A$(I)
80 NEXT I
100 M= N
105 REM   COMPARE FIRST HALF WITH SECOND HALF
110 M= INT (M/ 2)
120 IF M= 0THEN 275
130 J= 1
140 K= N- M
150 I= J
160 L= I+ M
170 IF A$(I)< A$(L)THEN 240
180 T$= A$(I)
190 A$(I)= A$(L)
200 A$(L)= T$
210 I= I- M
220 IF I< 1THEN 240
230 GOTO 160
240 J= J+ 1
250 IF J< = KTHEN 150
260 GOTO 110
275 FOR I= 1TO N
280 IF I< 10THEN PRINT " ";I;". ";
285 IF I> = 10THEN PRINT I;". ";
290 PRINT A$(I):NEXT I
300 DATA 10
305 DATA "ELEPHANT"
310 DATA "ZEBRA
315 DATA "CAMEL. CALVIN"
320 DATA "AARDVARD"
325 DATA "KANGAROO. KANGA"
330 DATA "DODGER. ROGER"
335 DATA "MILLICENT"
340 DATA "HIPPOPOTAMUS"
345 DATA "AUSSERORDENTLICHHOCHGESCHWINDIGKEITELEKTRONEN"
350 DATA "SNEEZE"

READY
```

BEST COPY AVAILABLE

22. B17a(j).2 - Quicksort sorting program. (page 1 of 2)

This is a Quicksort routine, as discussed in Supplement B17a(j), modified for alphabetization. The program was adapted from one appearing in Kilobaud Microcomputing, and is included here with permission of the publisher.

```
5 REM  PROGRAM FOR ALPHABETIZING A LIST OF N WORDS
7 REM   THE VALUE OF N SHOULD BE THE FIRST ITEM OF DATA. LINE 300

10 REM   THE SORTING PROCEDURE USED IS CALLED QUICKSORT
11 REM   THE HEART OF THE PROGRAM IS ADAPTED FROM A PROGRAM BY
12 REM   STEVEN HARRINGTON APPEARING IN THE APRIL 1979 ISSUE
13 REM   OF KILOBAUD MICROCOMPUTING. COPYRIGHT 1979. AND IS
14 REM   INCLUDED HERE WITH PERMISSION OF THE PUBLISHER.
15 CLEAR 1000:READ N
18 DIM A$(N).B(N)
20 FOR I= 1TO N
30 READ A$(I)
32 NEXT I
35 REM   INITIALIZE BEGINNING AND ENDING POINTS TO DO ENTIRE ARR
AY
40 L= 1
45 B(L)= N+ 1
50 M= 1
53 REM   SET END OF ARRAY SEGMENT
55 J= B(L)
58 REM   SET START OF ARRAY SEGMENT
60 I= M- 1
63 REM   IF ONLY 1 OR 2 ELEMENTS REMAIN HANDLE THEM SPECIALLY
65 IF J- M< 3THEN 165
70 M1= INT ((I+ J)/ 2)
73 REM   FIND A BIG ELEMENT AMONG THE SMALL ONES
75 I= I+ 1
80 IF I= JTHEN 125
85 IF A$(I)< = A$(M1)THEN 75
88 REM   FIND A SMALL ELEMENT AMONG THE BIG ONES
90 J= J- 1
95 IF I= JTHEN 125
100 IF A$(J)> = A$(M1)THEN 90
103 REM   EXCHANGE OUT OF PLACE ELEMENTS
105 T$= A$(I)
110 A$(I)= A$(J)
115 A$(J)= T$
120 GOTO 75
123 REM   ARRAY SEGMENT NOW DIVIDED
125 IF I> = M1THEN I= I- 1
130 IF J= M1THEN 150
135 T$= A$(I)
140 A$(I)= A$(M1)
145 A$(M1)= T$
148 REM   SAVE STARTING POINT FOR ARRAY SEGMENT OF BIG ELEMENTS
150 L= L+ 1
155 B(L)= I
158 REM   SAVE STARTING POINT FOR SMALL ARRAY
160 GOTO 55
165 IF J- M< 2THEN 190
170 IF A$(M)< A$(M+ 1)THEN 190
175 T$= A$(M)
180 A$(M)= A$(M+ 1)
185 A$(M+ 1)= T$
188 REM   SET POINTS FOR ARRAY SEGMENT OF BIG ELEMENTS
```

BEST COPY AVAILABLE

22. B17a(j).2 - Quicksort sorting program. (page 2 of 2)

```
190 M= B(L)+ 1
195 L= L- 1
200 IF L> 0THEN 55
220 FOR I= 1TO N
225 IF I< 10THEN PRINT " ";I;". ";
230 IF I> = 10THEN PRINT I;". ";
235 PRINT A$(I):NEXT 1
300 DATA 10
305 DATA "ELEPHANT"
310 DATA "ZEBRA
315 DATA "CAMEL, CALVIN"
320 DATA "AARDVARD"
325 DATA "KANGAROO, KANGA"
330 DATA "DODGER, ROGER"
335 DATA "MILLICENT"
340 DATA "HIPPOPOTAMUS"
345 DATA "AUSSERORDENTLICHHOCHGESCHWINDIGKEITELEKTRONEN"
350 DATA "SMEEZE"

READY

RUN

  1. AARDVARD
  2. AUSSERORDENTLICHHOCHGESCHWINDIGKEITELEKTRONEN
  3. CAMEL, CALVIN
  4. DODGER, ROGER
  5. ELEPHANT
  6. HIPPOPOTAMUS
  7. KANGAROO, KANGA
  8. MILLICENT
  9. SMEEZE
 10. ZEBRA


READY
```

BEST COPY AVAILABLE

Catalog of 16 mm. Computer Films

HISD AV Library

1. "Basic Computer Terms," 16 mm, M7560
2. "Career Awareness-Computer Industry," 11 min., 6980
3. "Computers," 11 min. 6065. Discusses components of a computer and library numbers.
4. "Data Processing - Introductory Principles," 14 min., M7189. Discusses use of punched cards.
5. "What is a Computer," 22 min., M5097. Discusses computer functions, terminology, and applications to space travel planning.

Filmstrips Available from the Mathematics Resource Center

1.	Working as a Computer Programmer	4489.018LA
2.	How a Computer Solves a Problem	4522.018DG
3.	Different Numeration Systems	4522.018G1N
4.	An Invitation to Computers	4529.019BFA

Also available at some schools:

5. How Computers Work, Encyclopedia Britannica Educational Corp., 1972

HOUSTON INDEPENDENT SCHOOL DISTRICT
INSTRUCTIONAL MEDIA SERVICES
PROFESSIONAL LIBRARY

BIBLIOGRAPHY: MICROCOMPUTERS

Adams, J. Mack. *Social Effects of Computer Use and Misuse*. John Wiley & Sons. 1976.

Ahl, David H., ed. *101 Basic Computer Games*. Digital Equipment Corp. 1974.

Ahl, David H. *Understanding Mathematics and Logic Using Basic Computer Games*. Digital Equipment Corp. 1973.

American Association of School Administrators. *Instructional Technology and the School Administrator*. AASA. 1970.

Andree, Richard V., Josephine P. Andree, and David D. Andree. *Computer Programming: Techniques, Analysis, and Mathematics*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey. 1973.

Artist and Computer. Ed. by Ruth Leavitt. Harmony Books. 1976.

Boillot, Michel. *Understanding Fortran*. West Publishing Co., St. Paul, Minn. 1978.

Bushnell, Don D., ed. *The Automation of School Information Systems*. Dept. of Audio-Visual Instruction of the National Education Association of the United States. 1964.

Barden, William. *How to Program Microcomputers*. Howard W. Sams & Co. 1977.

Barden, William. *Microcomputers for Business Applications*. Howard W. Sams & Co. 1979.

Bosworth, Bruce. *Programming in BASIC for Business*. Science Research Assoc. 1977.

Bushnell, Donald D. *The Computer in American Education*. Wiley. 1967.

The BYTE Book of Computer Music. Ed. by Christopher P. Morgan. BYTE Books. 1979.

Classroom Computer News. Miliken Publishing Co. Periodical beginning with March, 1981.

Cohen, Daniel. *Human Side of Computers*. McGraw. 1975.

The Computer in the School: Tutor, Tool, Tutee. Ed by Robert Taylor. Teachers College Press. 1980.

Computer Programming: BASIC for Microcomputers. Educational Activities, Inc. 1979. 5 Filmstrips and cassette tapes.

Computers and Education. Ed. by R.W. Gerard. McGraw-Hill. 1967.

Computers and Public Policy, proceedings of the symposium "Man and the Computer" held at Dartmouth College, November 30, December 1, 2, 1976.

Cox, Nigel S.M. *The Computer and the Library*. Archon Books. 1967.

Creative Computing. Periodical beginning with 1977.

Data Processing for Education Administration. American Data Processing. 1968.

Desmonde, William Herbert. *Computers and Their Uses*. Prentice-Hall. 1964.

Doyle, Lauren. *Information Retrieval and Processing*. Melville. 1975.

Dwyer, Thomas A. *Computer Resource Book - Algebra*. Houghton Mifflin. 1975.

Education, U.S.A. *Computers: New Era for Education?* National School Public Relations Association. 1968.

Entelek Incorporated. *Computer-Assisted Instruction Guide*. 1968.

Farrell, Edmund J. *English, Education, and the Electronic Revolution.* National Council of Teachers of English. 1967.

Frederick, Franz J. *Guide to Microcomputers*. Association for Educational Communications and Technology. 1980

Frenzel, Louis E. *The Howard W. Sams Crash Course in Microcomputers*. Howard W. Sams & Co.

Gruenberger, Fred. *Problems for Computer Solution*. Jonn Wiley & Sons. 1965.

Hickey, Albert E. *Computer-Assisted Instdruction: A Summary of Research in Selected Areas*. Entelek. 1975.

Hickey, Albert E. *The Logical Basis of Teaching*. Entelek Inc. 1964.

Hickey, Albert E. *Research Guidelines for Computer-assisted instruction*. Entelek. 1974.

Hoppock, Robert. *Occupational Information: Where to Get It and How to Use It In Career Counseling, and Career Development*. 4th ed. McGraw-Hill. 1976.

Lekan, Helen A. *Index to Computer Assisted Instruction*. 2nd ed. Sterling Institute. 1970.

Lien, David A. *The BASIC Handbook: An Encyclopedia of the BASIC Computer Language*. Compsoft Publishing. 1978.

Luce, Thomas G. *Genetics With the Computer*. Portsmouth, N.H. 1978.

Martin, James. *The Wired Society*. Prentice-Hali. 1978.

Mathews, Max V. *The Technology of Computer Music*. M.I.T. Press. 1969.

Moursund, David. *Calculators and Elementary Education*. Math Learning Center. 1979.

Mullish, Henry. *A Business-like Approach to COBOL*. Harper & Row Publishers, Inc. New York. 1979.

Orr, William D. *Conversational Computers*. Wiley. 1968.

Poole, Lon. *Accounts Payable & Accounts Receivable CBASIC*. Osborne/McGraw-Hill. 1979.

Poole, Lon. *General Ledger CBASIC*. Osborne/McGraw-Hill. 1979.

Poole, Lon. *Payroll With Cost Accounting CBASIC*. Osborne/McGraw-Hill. 1979.

Poole, Lon. *Some Common BASIC Programs*. 3rd ed. Osborne/McGraw-Hill. 1979.

Rader, Richard A. *Entelek Computer-Based Physics Lab*. Entelek. 1978.

Sage, Edwin R. *Problem-solving With the Computer*. Entelek, Inc. 1969.

Sippl, Charles J. *Computer Dictionary and Handbook*. H.W. Sams. 1966.

Soucek, Branko. *Computers in Neurobiology and Behavior*. John Wiley & Sons. 1976.

Spencer, Donald D. *Computers in Society: The Wheres, Why, and Hows of Computer Use*. Hayden Book Co., Inc. 1974.

U.S. Department of Health, Education and Welfare, Office of Education. *Computer-Based Vocational Guidance Systems*. HEW, Office of Education. 1969.

Uttal, William R. *Generative Computer-assisted Instruction in Analytic Geometry*. Entelek Inc. 1970.

Wessel, Milton R. *Freedom's Edge: The Computer Threat to Society*. Addison-Wesley Publishing Co. 1974.

END OF DOCUMENT